UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

PROYECTO FIN DE CARRERA

PROTOTIPADO DE UN SISTEMA WiMAX MIMO 2X2 (I) - EMISOR

INGENIERÍA DE TELECOMUNICACIÓN

Autor: ROBERTO PRIETO ALONSO

Tutores: Dr. VÍCTOR P. GIL JIMÉNEZ

Dr. ENRIQUE SAN MILLÁN HEREDIA

JULIO 2011

A todos los que han estado a mi lado
durante todos estos años…

A mis padres
A mis amigos y compañeros
.
… y muy especialmente a quien ya no está.

RESUMEN

El presente proyecto consiste en el diseño de un emisor para el estándar IEEE 802.16d-2004 utilizando MIMO 2x2, OFDM (Multiplexado por División en Frecuencias Ortogonales) en la capa física y BPSK (Modulación Binaria por Salto de Fase) para la modulación de datos, empleando una plataforma de desarrollo sobre FPGA basada en Simulink®, la cual proporciona un alto nivel de abstracción en el diseño de los modelos.

Todo ello, como respuesta al creciente interés que las comunicaciones inalámbricas de alta capacidad han suscitado más allá, incluso, del mundo de las telecomunicaciones.

El proyecto busca, así mismo, adaptar a dicho interés técnicas importantes como son las de diversidad MIMO (Múltiples Entradas Múltiples Salidas) y tecnologías relevantes y de gran potencial como son las plataformas de desarrollo basadas en FPGA (Matrices de Puertas Programables)

Los últimos años han sido testigos del desarrollo y despliegue de capacidades de comunicación inalámbrica de alta capacidad. Estos desarrollos han seguido dos caminos diferenciados, en función de la tipología de cobertura, siendo dominante la familia de estándares IEEE 802.11 (Wi-Fi, Fidelidad Inalámbrica) para redes LAN, típicamente privadas, y los sistemas HSDPA (Acceso Descendente de Paquetes a Alta Velocidad) introducidos por el 3GPP (Proyecto Conjunto de Tercera Generación), para las redes de telefonía móvil.

Sin embargo, dos nuevas familias de estándares pueden suponer una revolución en el mercado de las comunicaciones inalámbricas de alta capacidad. Estos son IEEE 802.11d-2004 (WiMAX Fijo, significando WiMAX Interoperabilidad Mundial para Acceso por Microondas) e IEEE 802.16e-2005 (WiMAX Móvil) por un lado, e IEEE 802.20 (Mobile-Fi, Fidelidad Móvil) por el otro. Si bien ambos cuentan con características similares, WiMAX Móvil cuenta con mayor experiencia de mercado y apoyo por parte de la industria, destacando el apoyo de grandes fabricantes, como Intel, Alcatel-Lucent o Nokia y operadores, como, British Telecom, Deutsche Telecom, y Vodafone.

Las técnicas MIMO mejoran las prestaciones sobre el canal de comunicación, incrementando el radio de la celda y dotándola de mayor capacidad. Es por ello una de las técnicas fundamentales en los últimos estándares desarrollados y en los proyectados a futuro.

Por su parte, las plataformas de desarrollo basadas en FPGA han demostrado en los últimos años una mejora en la velocidad de proceso y capacidad de almacenamiento, lo que le ha permitido adentrarse en campos de aplicación típicos de los ASIC (Circuitos Integrados para Aplicaciones Específicas), que cuenta con la desventaja de no ser reprogramables. Así mismo,

durante los últimos años han aparecido entornos de desarrollo cada vez más sencillos y flexibles.

Este proyecto cuenta con el objetivo principal de aunar los tres puntos anteriores: IEEE 802.16d-2004 (WiMAX Fijo), técnicas MIMO y desarrollo sobre FPGA.

ABSTRACT

The present project consists in the designing of a transmitter for IEEE 802.16e-2005 standard using 2x2 MIMO, OFDM (Orthogonal Frequency Division Multiplexing) at the physical layer and BPSK (Binary Phase-Shift Keying) for data modulation, using an FPGA development platform based on Simulink ®, which provides a high level of abstraction while designing models.

The Project has been developed in response to the growing interest in the high-capacity wireless communications that have risen in the telecommunications sector and further.

Likewise, the project aims at adapting major interesting techniques, as MIMO diversity (Multiple Input Multiple Output) and relevant technologies with great potential such as the development platforms based on FPGA (Programmable Gate Arrays)

Recent years have witnessed the development and deployment of high-capacity wireless communication capabilities. These developments has followed two distinct paths, depending on the type of coverage, being two families of standards market-dominant, IEEE 802.11 (Wi-Fi, Wireless Fidelity) for typically private LANs, and HSDPA systems (Packet Access High Descending speed) introduced by the 3GPP (Third Generation Partnership Project) for mobile phone networks.

However, two new families of standards may mean a revolution in the market for high capacity wireless communications. These are IEEE 802.11d-2004 (Fixed WiMAX, meaning WiMAX Worldwide Interoperability for Microwave Access) and IEEE 802.16e-2005 (Mobile WiMAX) on one side and IEEE 802.20 (Mobile-Fi, Mobile Fidelity) on the other. While both have similar characteristics, Mobile WiMAX has more market experience and support from the industry, highlighting among supporters manufacturers as Intel, Alcatel-Lucent and Nokia, and operators like British Telecom, Deutsche Telecom and Vodafone.

MIMO techniques improve the performance over the communication channel, increasing the radius of the cell and giving it greater capacity. That is why it is now one of the fundamental techniques on the latest developed standards and the ones to come in the recent future.

For its part, the development platforms based on FPGA have demonstrated in recent years an improvement in processing speed and storage capacity, which has allowed them to enter in typical application fields of ASIC (Application Specific Integrated Circuits), which have the

disadvantage of not being reprogrammable. Also, recent years have seen development environments more and more simple and flexible.

This project has the main objective of joining the three points above related: IEEE 802.16e-2005 (Mobile WiMAX), and MIMO techniques on FPGA development

GLOSARIO

3GPP	3rd Generation Partnership Project (Proyecto Conjunto de Tercera Generación)
AAS	Adaptative Antenna System (Sistema de Antena Adaptativa)
ACI	Adjacent Channel Interference (Interferencia de Canal Adyacente)
AES	Advanced Encryption Standard (Estándar Avanzado de Encriptación)
ADC	Analog to Digital Converter (Conversor Analógico Digital)
ADOA	Amplitude Difference Of Arrival (Diferencia en Amplitud de Llegada)
All-IP	All Internet Protocol (Todo Protocolo de Interconexión de Redes)
ASIC	Application Specific Integrated Circuit (Circuito Integrado para Aplicaciones Específicas)
ATM	Asynchronous Transfer Mode (Modo de Transferencia Asíncrono)
BER	Bit Error Rate (Tasa de Error de Bit)
BPSK	Binary Phase Shift Keying (Modulación Binaria por Salto de Fase)
BS	Base Station (Estación Base)
CID	Connection Identifier (Identificador de Conexión)
CLB	Configurable Logic Block (Bloque Lógico Configurable)
CP	Cyclic Prefix (Prefijo Cíclico)
cPCI	Compact Peripheral Component Interconnect (Interconexión de Componentes Periféricos Compactos)
CPE	Customer Premises Equipment (Equipo Local del Cliente)
CPLD	Complex Programmable Logic Device (Dispositivo Lógico Programable Complejo)
DAC	Digital to Analog Converter (Conversor Digital Analógico)
DCM	Digital Clock Manager (Gestor del Reloj Digital)
DDR	Double Data Rate (Tasa Doble de Transferencia de datos)
DES	Data Encryption Standard (Estándar de Encriptación de Datos)
DFT	Discrete Fourier Transform (Transformada Discreta de Fourier)
DOA	Direction Of Arrival (Dirección de Llegada)
DSP	Digital Signal Processor (Procesador Digital de Señales)
FBWA	Fixed Broadband Wireless Access (Acceso Inalámbrico de Banda Ancha Fijo)
FDD	Frequency Division Duplex (Duplexado por División en Frecuencia)
FFT	Fast Fourier Transform (Transformada Rápida de Fourier)
FIFO	First In, First Out (Primero en Entrar, Primero en Salir)
FIR	Finite Impulse Response (Respuesta al Impulso Finita)
FPGA	Field Programmable Gate Array (Matriz de Puertas Programables)
GBps	Gigabyte Per Second (Gigabyte Por Segundo)

GMACS	Giga Multiply Accumulate Operations per Second (Gigaoperaciones Acumuladas de Multiplicación Por Segundo)
GSM	Global System for Mobile communications (Sistema Global para las Comunicaciones Móviles)
HDL	Hardware Description Language (Lenguaje de Descripción de Soportes Físicos)
HFDD	Half-Duplex Frequency Division Duplex (Duplexado Mitad por División en Frecuencia)
HSDPA	High Speed Downlink Packet Access (Acceso Descendente de Paquetes a Alta Velocidad)
HSUPA	High Speed Uplink Packet Access o (Acceso Ascendente de Paquetes a Alta Velocidad)
IFFT	Inverse Fast Fourier Transform (Transformada Rápida Inversa de Fourier)
IOB	Input Output Banks (Bancos de Entrada Salida)
IP	Internet Protocol (Protocolo de Interconexión de Redes)
ISI	Inter-Symbols Interference (Interferencia entre Símbolos)
ISP	Internet Service Provider (Proveedor de Servicios de Internet)
LAN	Local Area Network (Red de Área Local)
LOS	Line Of Sight (Línea de Visión Directa)
LTE	Long Term Evolution (Evolución a Largo Plazo)
MAC	Medium Access Control (Control de Acceso al Medio)
MAN	Metropolitan Area Network (Red de Área Metropolitana)
MB	Megabyte (Megabyte)
MBWA	Mobile Broadband Wireless Access (Acceso Inalámbrico de Banda Ancha Móvil)
MIMO	Multiple Input Multiple Output (Múltiples Entradas Múltiples Salidas)
MMCX	Micro-Miniature Coaxial (Coaxial Micro Miniaturizado)
Mobile-Fi	Mobile Fidelity (Fidelidad Móvil)
NAS	Network Attached Storage (Almacenamiento Conectado a Red)
NLOS	Non Line Of Sight (Sin Línea de Visión Directa)
OFDM	Orthogonal Frequency Division Multiplexing (Multiplexado por División en Frecuencias Ortogonales)
OFDMA	Orthogonal Frequency Division Multiple Access (Acceso Múltiple por División en Frecuencias Ortogonales)
PHY	Physical Layer (Capa Física)
QAM	Quadrature Amplitude Modulation (Modulación en Amplitud por Cuadratura)
QoS	Quality of Service (Calidad de Servicio)
QPSK	Quadrature Phase Shift Keying (Modulación en Cuadratura por Salto de Fase)

RAM	Random Access Memory (Memoria de Acceso Aleatorio)
RF	Radio Frequency (Radio Frecuencia)
ROM	Read Only Memory (Memoria de Sólo Lectura)
SAN	Storage Area Network (Red de Área de Almacenamiento)
SDR	Software-Defined Radio (Equipos de Radio Definidos por Software)
SDRAM	Synchronous Dynamic Random Access Memory (Memoria Dinámica de Acceso Aleatorio Síncrona)
SFID	Service Flow Identifier (Identificador de Flujo de Servicio)
SRAM	Static Random Access Memory (Memoria Estática de Acceso Aleatorio)
SS	Subscriber Station (Estación Suscriptora)
STC	Space Time Coding (Codificación Espacio Tiempo)
TDD	Time Division Duplex (Duplexado por División en Tiempo)
TDM	Time Division Multiplexing (Multiplexado por División en Tiempo)
TDOA	Time Difference Of Arrival (Diferencia en Tiempo de Llegada)
UL	Uplink (Enlace Ascendente)
UMTS	Universal Mobile Telecomunication System (Sistema Universal de Telecomunicaciones Móviles)
VHDL	Very High Speed Integrated Circuit Hardware Description Language (Lenguaje de Descripción de Soportes Físicos para Circuitos Integramos Muy Rápidos)
VLAN	Virtual Local Area Network (Red de Área Local Virtual)
VoIP	Voice over IP (Voz Sobre IP)
WAN	Wide Area Network (Red de Área Amplia)
WCDMA	Wideband Code Division Multiple Access (Acceso múltiple por División de Código de Banda Ancha)
WiBro	Wireless Broadband (Banda Ancha Inalámbrica)
Wi-Fi	Wireless Fidelity (Fidelidad Inalámbrica)
WiMAX	Worldwide Interoperability for Microwave Access (Interoperabilidad Mundial para Acceso por Microondas)
xDSL	Digital Subscriber Line (Línea de Suscripción Digital)

INDICE

1. INTRODUCCIÓN

Los últimos años han sido testigos de un rápido crecimiento en el empelo comercial de tecnologías de comunicaciones inalámbricas, junto con la oferta de servicios y contenidos digitales que requieren cada vez mayores anchos de banda.

La fuerte demanda experimentada ha supuesto un caldo de cultivo excepcional para el desarrollo de nuevas tecnologías y técnicas, capaces de satisfacer las crecientes necesidades del mercado, como las técnicas MIMO (Múltiples Entradas Múltiples Salidas).

Asimismo, la realización del presente proyecto fue posible gracias al gran potencial que presentan las plataformas de desarrollo basadas en FPGA (Matrices de Puertas Programables)

1.1 Antecedentes del proyecto

A lo largo del año 2008 el autor del presente proyecto (Roberto Prieto Alonso) colaboró con David Díaz Martín en la realización de un proyecto conjunto, consistente en el diseño, simulación y validación de un sistema de comunicación para el estándar IEEE 802.16d-2004, mediante técnicas MIMO 2x2 y OFDM (Multiplexado por División en Frecuencias Ortogonales) para la capa física y modulación de datos mediante BPSK (Modulación Binaria por Cambio de Fase).

Para su realización se empleó una plataforma de desarrollo para FPGA basada en Simulink® denominada VHS-ADC/DAC-Virtex-4 de Lyrtech, junto con *Xilinx® System Generator for DSP* (Procesador Digital de Señales).

Como paso previo al proyecto, los autores elaboraron un Estudio Tecnológico, titulado “Estudio práctico sobre el prototipado de un sistema MIMO 2x2 para WiMAX” (Interoperabilidad Mundial para Acceso por Microondas).

Dicho estudio buscaba, en primer lugar, lograr una caracterización de la plataforma de desarrollo sobre FPGA ya comentada. Para ello se elaboraron, diseñaron y sintetizaron, sencillos modelos mediante la herramienta *Xilinx® System Generator* que, ejecutados y validados, permitieron comprender y caracterizar el funcionamiento de la plataforma.

En segundo lugar, el estudio consistió en la elaboración, sobre la misma plataforma, de un modelo de comunicaciones básico que sirviese como semilla de un diseño para el estándar IEEE 802.16e-2005 (WiMAX Móvil). Dicho modelo, en el que destaca el empleo de un esquema IFFT/FFT (Transformada Rápida de Fourier Inversa / Transformada Rápida de Fourier),

consistía en un receptor y un emisor que fueron diseñados, simulados y validados por los autores.

El presente proyecto describirá el emisor elaborado dentro del proyecto conjunto descrito, que se desarrolla alrededor del modelo básico diseñado anteriormente y haciendo uso del conocimiento obtenido en el Estudio Tecnológico.

1.2 Objetivos del Proyecto

Los objetivos del proyecto pueden resumirse como sigue:

1. Desde el punto de vista del estándar IEEE 802.16d-2004 (WiMAX Fijo): estudiar y analizar el estándar en su definición OFDM para la capa física.

2. Desde el punto de vista de las técnicas MIMO 2x2: estudiar y analizar las mismas en referencia al estándar IEEE 802.16d-2004 (WiMAX Fijo)

3. Desde el punto de vista de la plataforma de desarrollo sobre FPGA basada en Simulink®: diseñar, simular y validar un emisor en línea con los objetivos anteriores, esto es, empleando MIMO 2x2 y OFDM para el estándar IEEE 802.16d-2004 (WiMAX Fijo)

El presente proyecto reflejará el conocimiento obtenido mediante la caracterización de la plataforma FPGA basada en Simulink®: empleada (VHS-ADC/DAC Virtex-4 de Lyrtech junto a la herramienta Xilinx System Generator for DSP), prestando especial atención a los bloques empleados en el diseño y las limitaciones detectadas en los mismos.

Cabe destacar que el presente proyecto cubre parcialmente el ciclo completo de implementación del sistema final sobre placa FPGA, que se pueden resumir en los siguientes puntos.

1. Caracterización de la plataforma tecnológica empleada.(Ver apartado 3 y Anexos)
    - 1.1. Elementos Hardware (componentes e interfaces): Lyrtech Virtex.
    - 1.2. Elementos Software (programas, versiones y licencias): Xilinx ®, Matlab ®, Simulink ®
    - 1.3. Relaciones Hardware/Software (interfaces, captura e inyección de datos, etc.)
    - 1.4. Caracterización de los elementos de diseño (bloques, etc.)
2. Diseño del sistema
    - 2.1. Análisis de requerimientos: estándares y especificaciones. (Ver apartado 2)
    - 2.2. Diseño con Simulink (utilizando bibliotecas de Xilinx® System Generator y Lyrtech). (Ver apartado 4)
    - 2.3. Simulación con Simulink.(Ver apartado 5)

3. Síntesis y simulación
    3.1. Síntesis con Xilinx® System Generator
    3.2. Síntesis más detallada con ISE de Xilinx (partiendo de los ficheros generados en el punto 2.3.)
    3.3. Simulación funcional con ISE de Xilinx/Modelsim (partiendo de los ficheros generados en el punto 2.3.)
    3.4. Simulación post place and route con ISE de Xilinx (partiendo de los ficheros generados en los puntos 2.3 y 3.1)

El presente proyecto cubre los puntos 1 y 2 mencionados. Además, a efectos de verificación, se realizó una síntesis de un modelo debidamente preparado para ello (modelo conjunto emisor-receptor), si bien este último aspecto queda fuera del ámbito del presente documento.

1.3 Descripción de contenidos

Al objeto de alcanzar los objetivos marcados en el punto anterior, el presente proyecto cubrirá los siguientes aspectos.

1. **Descripción del estándar IEEE 802.16 (WiMAX).**

    Se realizará una exposición de dicha tecnología, cubriendo sus principales aplicaciones y se presentarán las principales versiones existentes del estándar:

    1.1. IEEE 802.16d-2004 - WiMAX Fijo

    1.2. IEEE 802.16e-2005 -WiMAX Móvil

    1.3. IEEE 802.16e-2005 – WiBro

    Se realizará una comparativa entre el estándar empleado en el proyecto, IEEE 802.16e-2005 (WiMAX Móvil), con otras tecnologías de comunicaciones inalámbricas existentes en el mercado:

    1.4. Wi-Fi (Fidelidad Inalámbrica)

    1.5. Mobile-Fi (Fidelidad Inalámbrica)

    1.6. 3G HSDPA (Tercera Generación Acceso por Paquetes a Enlace Descendente de Alta Velocidad)

    Se expondrán las primeras experiencias de implantación en España de WiMAX.

2. **Caracterización de la plataforma empleada**.

Se describirán y caracterizarán de forma detallada los bloques que se emplean en el emisor diseñado.

3. **Diseño del emisor**

   Se expondrán cada una de las etapas del emisor, en base a los requisitos que establece el estándar, junto con la debida justificación de las simplificaciones asumidas.

4. **Proceso de Validación: Simulación y Resultados**

   Se mostrarán los resultados de simulación mediante los flujos de datos en distintos puntos del sistema.

   Dado que el emisor objeto del presente proyecto se realizó en el marco de un proyecto conjunto con el correspondiente receptor, se aprovechará dicha circunstancia para mostrar las señales del emisor junto a sus pares en el receptor, al objeto de demostrar correcto funcionamiento del sistema al completo.

5. **Conclusiones y trabajos futuros**.

   Se extraerán las principales conclusiones fruto del presente trabajo y se propondrán posibles mejoras al sistema, así como potenciales líneas de trabajo que se puedan desarrollar como continuación al proyecto.

6. **Presupuesto**

   Se detallará el coste de elaboración del presente proyecto, buscando la exhaustividad en la localización de sus fuentes.

7. **Bibliografía**

   Se documentarán las fuentes empleadas en la elaboración de este proyecto.

8. **Anexos**.

   Se aportará información ampliada referente a varios aspectos del proyecto: código de las funciones MATLAB® implementadas y fundamentos de FPGA, MATLAB® y Simulink®.

2. ESTANDAR IEEE 802.16 (WIMAX)

El WiMAX fórum es una organización privada sin ánimo de lucro y dirigida por la industria, cuya misión es certificar y promover la interoperabilidad de productos inalámbricos de banda ancha, basados en los estándares armonizados IEEE 802.16/ETSI HiperMAN

Fruto de su actividad surgen una serie de estándares inter operables, entre los que destacan, al objeto del presente proyecto, el estándar IEEE 802.16d-2004 y el estándar IEEE 802.16e-2005. Este último, IEEE 802.16e-2005 es conocido como WiMAX Móvil, es sobre el que se implementa el presente proyecto.

WiMAX Forum define perfiles que fijan las distintas opciones que dan lugar a la familia de estándares, como modulación, ancho de banda, frecuencia de utilización. Adicionalmente, los complementa en áreas no cubiertas, como arquitectura o sistemas de prueba (ver figura 1).

Figura 1. *: Definición del perfil Mobile WiMAX Release 1. Fuente: Fundación Telefónica e Intel*

La familia de estándares de WiMAX definen la capa MAC (Control de Acceso al Medio) y la capa PHY (Capa Física), creando una única torre de distribución que permite conexiones inalámbricas a velocidades similares a las que se puede obtener empleando tecnologías xDSL (Bucle de abonado Digital) en entornos Sin Línea de Visión Directa (NLOS). Los estándares IEEE 802.16d-2004 e IEEE 802.16e-2005 poseen una Línea de Visión Directa de 50 kilómetros y 10 kilómetros respectivamente (ver figura 2).

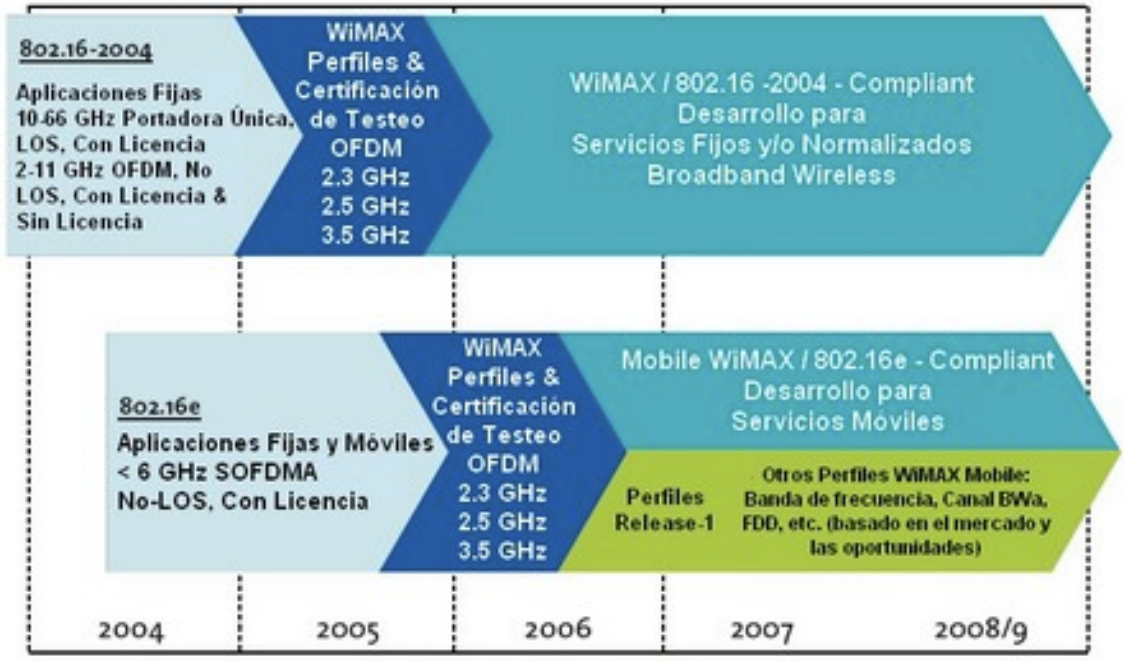

Figura 2. Evolución de la tecnología WiMAX. Fuente: Fundación Telefónica

La familia de estándares WiMAX surge para cubrir las necesidades de transmisión inalámbrica de alto ancho de banda. En la actualidad, la tecnología celular basada en HSDPA (Acceso Descendente de Paquetes a Alta Velocidad) sobre 3G, no es capaz de proveer de anchos de banda elevados a un gran número de usuarios, lo que dificulta el acceso a contenidos multimedia.

Las tecnologías WiMAX suponen una alternativa viable para desplegar redes de acceso de gran ancho de banda en zonas no cubiertas por otras tecnologías. En las zonas rurales, el coste de cobertura de la última milla dificulta el despliegue de tecnologías por cable o fibra óptica. En países en vías de desarrollo suponen una alternativa al despliegue de redes móviles 3G. Las tecnologías WiMAX permiten llevar banda ancha a un gran número de usuarios con un menor coste de implantación que otras tecnologías, cubriendo grandes áreas, sin necesidad de Línea de Visión Directa.

WiMAX destaca por su capacidad como tecnología portadora. Permite transportar servicios con protocolos por paquetes como IP (Protocolo de Interconexión de Redes), servicios conmutados como TDM (Multiplexado por División en el Tiempo), E1/T1, interconexiones ATM (Modo de Transferencia Asíncrono) y Frame Relay. WiMAX soporta múltiples de estos servicios simultáneamente, definiendo términos QoS (Calidad de Servicio) propios. Esta versatilidad que permite, por ejemplo, transmitir VoIP (Voz sobre IP) y voz tradicional (Clase-5), hace esta tecnología atractiva tanto para los operadores de telecomunicaciones como para los propietarios de grandes redes corporativas de voz y datos.

WiMAX Forum e Intel constituyen el núcleo institucional detrás del impulso de WIMAX, actuando el segundo como principal propulsor industrial de la tecnología. Tal y como se explica al principio de este apartado, WiMAX Forum fomenta la interoperabilidad entre las diferentes marcas de productos. WiMAX Forum realiza procedimientos de certificación en laboratorios oficiales, tras el cual expide el certificado. En la actualidad, posee laboratorios en Estados Unidos, España, China, Corea del Sur, China y Taiwan (2) y planea abrir dos más, en Brasil y Malasia (ver figura 3).

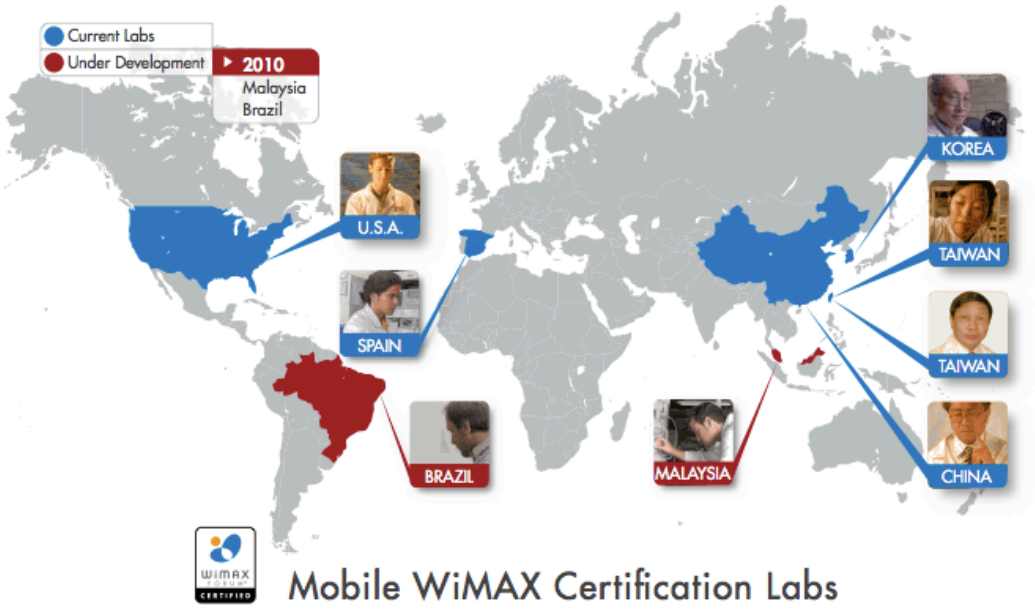

Figura 3. Ubicación de los laboratorios de certificación. Fuente: Intel

Además de Intel, otras muchas compañías forman parte del WiMAX Forum. Entre ellas, podemos destacar: Acer, Agilent Technologies, Alcatel-Lucent, Alvarion, AT&T, British Telecom, Cisco Systems, Dell, Deutsche Telecom AG, EADS Secure Networks, Ericsson, France Telecom, Fujitsu, Google, HTC Corporation, Intel Corporation, Microsoft Corporation, Motorola, NTT (Nippon Telegraph and Telephone), Nokia, Nortel, RIM (Research in Motion Limited), Samsung, Siemens AG, Sony Electronics, Symbian, Texas Instruments, Toshiba, VeriSign, Vodafone, Xilinx, …

WiMAX Forum se encuentra integrada por más de 500 organizaciones. Además de las compañías mencionadas, WiMAX Forum colabora con los gobiernos mundiales en asuntos relativos a la asignación de espectros, aspecto fundamental para el despliegue de la tecnología.

A continuación se describen en detalle aspectos de la tecnología y se compara con otras tecnologías existentes con las que WiMAX deberá coexistir.

2.1 Principales versiones: características y aplicaciones

WiMAX surge alrededor de una serie de estándares con características y aplicaciones diversas (se muestran en la tabla 1).

Tabla 1. Familia de estándares IEE 802.16, incluyendo WiBro, a nivel físico

	IEE				
	802.16	802.16a	802.16d-2004 Imax Fijo	802.16e-2005 WiMAX Móvil	802.16e-2005 WiBro
Fecha de aprobación	Diciembre 2001	Enero 2003	Junio 2004	Diciembre 2005	Noviembre 2004
Espectro (GHz.)	10 - 66	2 -11	2 - 11	2 - 6	2.3 GHz – 2.4 GHz (Corea, Europa y EEUU).
Funcionamiento	Visión directa (LOS)	Sin visión directa (NLOS)	Sin visión directa (NLOS)	Sin visión directa (NLOS)	Sin visión directa (NLOS)
Tasa de bit	Hasta 134 Mbit/s con canales de 28 MHz	Hasta 75 Mbit/s con canales de 20 MHz	Hasta 75 Mbit/s con canales de 20 MHz	Hasta 15 Mbit/s con canales de 5 MHz	Hasta 30 Mbit/s
Modulación	QPSK, 16QAM y 64 QAM	OFDM con 256 subportadoras QPSK, 16QAM, 64QAM	OFDM con 256 subportadoras BPSK, QPSK, 16QAM, 64QAM	OFDMA con hasta 2048 subportadoras BPSK, QPSK, 16QAM, 64QAM	OFDMA con 1024 subportadoras BPSK, QPSK, 16QAM, 64QAM
Movilidad	NO	NO	NO	SI (hasta 120 km/h)	SI (hasta 120 km/h)
Anchos de banda	20, 25 y 28 MHz	Seleccionables entre 1,25 y 20 MHz	Seleccionables entre 1,25 y 20 MHz	Seleccionables entre 1,25 y 20 MHz	9 MHz
Radio de celda típico	2 - 5 km	5 - 10 km aprox. (alcance máximo de unos 50 km con LOS)	5 - 10 km aprox. (alcance máximo de unos 50 km con LOS)	2 - 5 km aprox. (alcance máximo de unos 10 km con LOS)	2 km

En la tabla anterior se marcan en fondo gris las dos principales versiones de WiMAX:

WiMAX Fijo - IEEE 802.16d-2004

WiMAX Móvil – IEEE 802.16e-2005

La principal diferencia entre ambas radica en que el primero es un estándar inalámbrico fijo y el segundo añade el concepto de movilidad de los terminales.

2.1.1 WiMAX Fijo (IEEE 802.16d-2004)

Características.

IEEE 802.16d-2004, cuyas principales características se describen en la Tabla 1, emplea OFDM para ofrecer un funcionamiento sin línea de vista directa (NLOS) de varios kilómetros. El estándar permite servir a varios usuarios creando en ellos la sensación de que están transmitiendo y recibiendo continuamente.

A nivel físico, WiMAX Fijo permite tanto funcionamiento TDD (Duplexado por División en el Tiempo) y FDD (Duplexado por División en Frecuencia) en la BS (estación base). En la SS (estación suscriptora) se emplea HFDD (Sema-Duplexado por División en Frecuencia).

A nivel de Control de Acceso al Medio (MAC), el estándar ha sido definido para acomodar múltiples protocolos, tanto existentes (ATM, Ethernet e IP) como otros que puedan surgir en el futuro. Adicionalmente, se pueden definir niveles de calidad de servicio (QoS) para los niveles superiores. La máxima eficiencia en el empleo del ancho de banda del canal ascendente (UL) se obtiene mediante asignación dinámica de slots de frecuencia.

IEEE 802.16d-2004 está orientado a conexión. Cada estación suscriptora tiene un CID (identificador de Conexión) único. Existen CID básicos, primario, de transporte y de difusión. La seguridad la proporcionan el protocolo PKM, para la gestión de claves, empleando certificados digitales X.509, y el algoritmo RSA en el cifrado. También es posible emplear algoritmos de cifrado simétrico: DES (Estándar de Encriptación de Datos) y AES (Estándar Avanzado de Encriptación)

Aplicaciones

WiMAX Fijo supone una alternativa a las conexiones por cable y xDSL. Permite proporcionar servicios de voz (preferiblemente sobre IP) y banda ancha en regiones donde el despliegue de otras tecnologías “terrestres”, como regiones rurales donde la dispersión de los puntos finales de servicio, hace inviable el tendido de cable o fibra. En la localización del cliente se instalaría un CPE (Equipo en el recinto del Cliente), que se conectaría a la red WiMAX y daría servicio a la vivienda o empresa. Adicionalmente, WiMAX fijo permite su empleo como Backhaul inalámbrico (interconexión de redes).

Tal y como se muestra en la figura siguiente (figura 4), la red WiMAX puede dar servicio a múltiples localizaciones permitiendo interactuar con otras tecnologías.

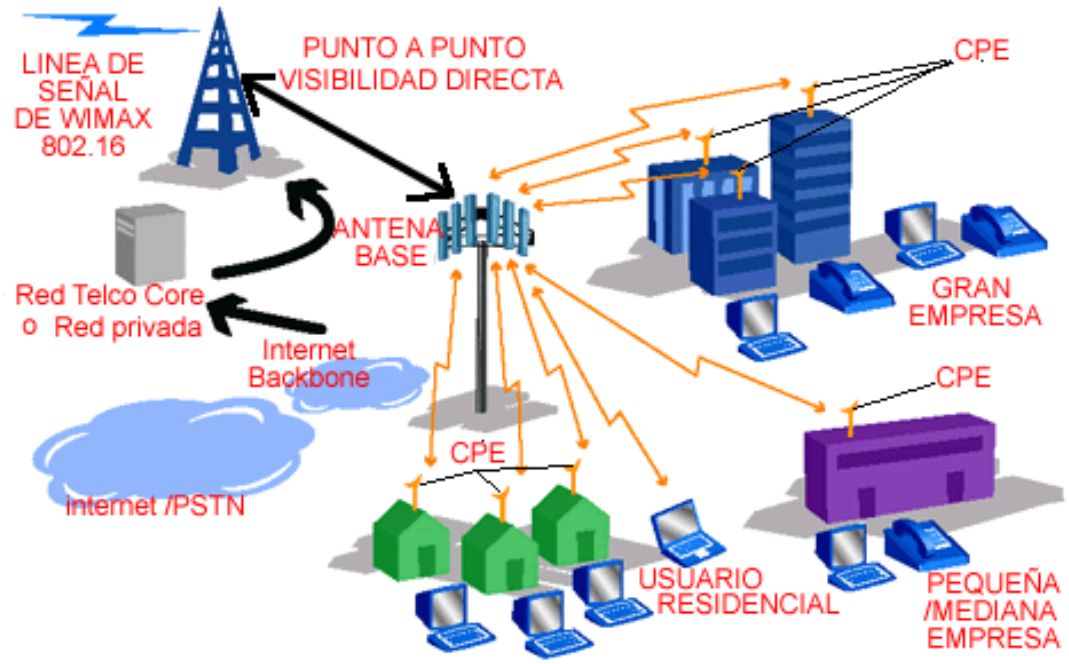

Figura 4. Red WiMAX Fijo. Fuente: Observatorio Tecnológico del Ministerio de Educación

2.1.2 WiMAX Móvil (IEEE 802.16e-2005)

Características.

El estándar IEEE 802.16e-2005, WiMAX Móvil, supone una ampliación de las posibilidades anteriores, incluyéndose conceptos de movilidad e intinerancia.

WiMAX Móvil incorpora OFDMA (Acceso múltiple por División en Frecuencias Ortogonales), que mejora la asignación de ancho de banda. Para permitir su empleo en comunicación móvil en entornos con fuertes interferencias, obstáculos y multitrayectos se emplean antenas inteligentes MIMO y AAS (Sistemas de Antenas Adaptativas).

A nivel físico, WiMAX Móvil emplea las mismas modulaciones que las descritas para WiMAX fijo. A nivel MAC, se mantienen también las características anteriores, con aspectos propios de la movilidad, soportando tráfico continuo y a ráfagas.

WiMAX sustituye el protocolo de gestión de claves PKM por el PKM-II

Aplicaciones

Una red WiMAX basada en IEEE 802.16e-2005 podría desplegarse de forma similar a una red de telefonía celular. De esta forma, la región a cubrir se dividiría en una serie de áreas solapadas, celdas. Cada celda da servicio a los usuarios que se encuentran en ella. Se implementan procedimientos de traspaso de las conexiones entre celdas de forma transparente al usuario, de forma que este recibe servicio en movilidad dentro de toda la región cubierta.

Sin embargo, este tipo de despliegues requieren el empleo de bandas de frecuencia sujetas a licencia, a fin de reducir interferencias, por lo que el despliegue esta sujeto a los marcos regulatorios estatales y los sobre-costes asociados a la obtención de licencias.

Además de las aplicaciones heredadas de WiMAX Fijo, WiMAX Móvil permite pensar en aplicaciones propias de la movilidad, como telefonía celular, ancho de banda móvil, servicios basados en localización, etc. (ver figura 5).

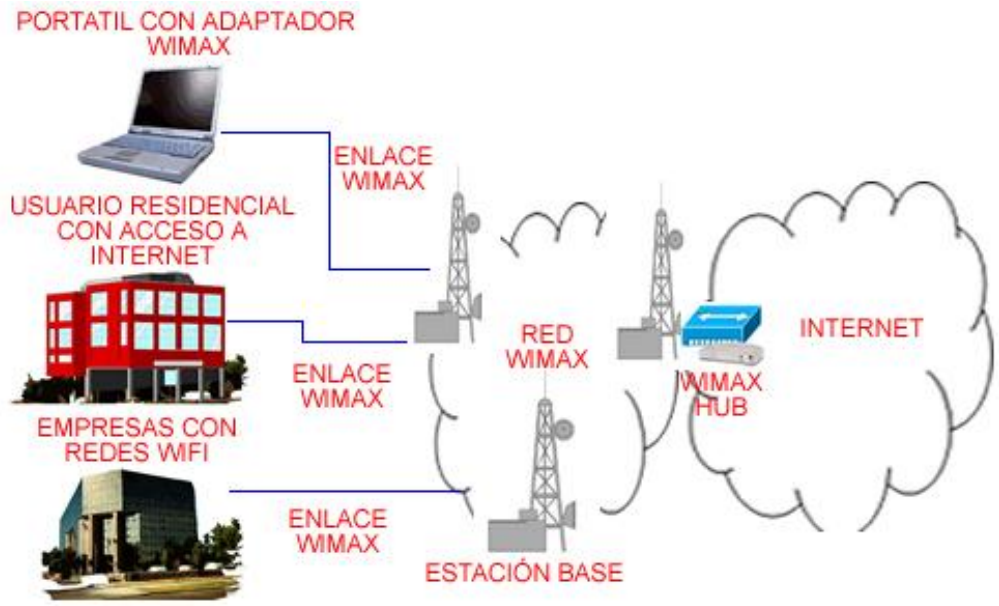

Figura 5. . Red WiMAX Fijo. Fuente: Observatorio Tecnológico del Ministerio de Educación

2.1.3 WiBro (IEEE 802.16e-2005)

WiBro (Banda Ancha inalámbrica) es un estándar creado en Corea del Sur a finales de 2004. Su objetivo es prestar acceso inalámbrico de banda ancha a dispositivos fijos y móviles. El estándar se basa en WiMAX Móvil. Se creación fue apoyada por WiMAX Forum e INTEL, siendo reconocido como un perfil de WiMAX Móvil.

Para maximizar el número de usuarios por celda y la eficiencia espectral, WiBro emplea modulación TDD

2.2 Comparativa de WinFax Móvil (802-16e-2005) con otras tecnologías.

A la vista de la descripción anterior y teniendo en cuenta las tecnologías sobre las que se fundamenta WiMAX, OFDM, MIMO, All-IP (Todo sobre IP), WiMAX se postula como una alternativa viable para proporcionar acceso con muy alto ancho de banda de forma inalámbrica frente a los dos tecnologías básicas empleadas comercialmente en la actualidad (ver figura 6):

- WI-FI: sobre la que WiMAX ofrece mayor cobertura y movilidad
- 3G HSDPA: sobre la que WiMAX ofrece mayores tasas binarias.

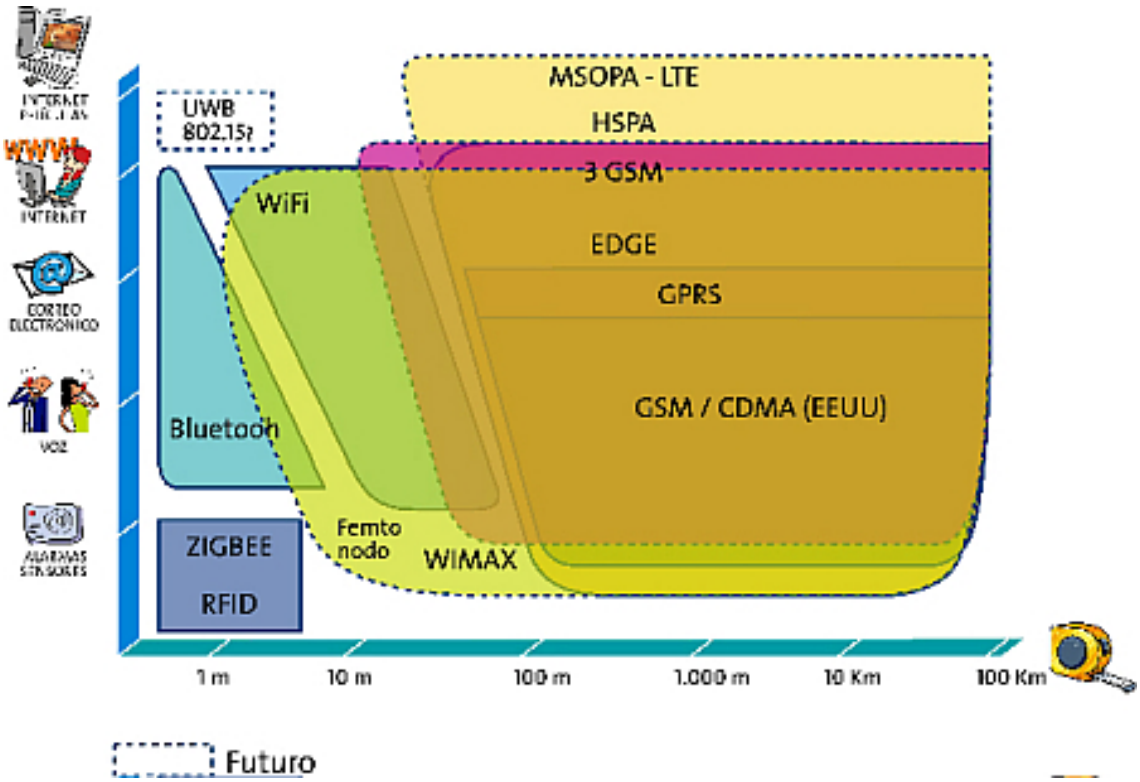

Figura 6. Cobertura y Capacidad de tecnologías inalámbricas. Fuente: Fundación Telefónica

2.2.1 WiMAX Móvil frente 3G HSDPA

El sistema de telefonía móvil global 3G es promovido por el 3GPP (Proyecto Conjunto de Tercera Generación) y supone la evolución de la telefonía móvil desde el estándar GSM. Para mejorar sus prestaciones, se incorpora HSDPA, un avance en WCDMA (Acceso Múltiple por División de Código de Banda Ancha), que permite aumentar las velocidades de transmisión de los equipos.

3GPP identifica tres fases en la evolución de HSDPA:

1. HSDPA básico. Definido en la *Reléase* 5. Tasas de entre 10.8 Mbit/s y 14.4 Mbit/s

2. Se agregan HSUPA (Acceso Ascendente de Paquetes a Alta Velocidad) y antenas inteligentes

3. Se combinan OFDM y MIMO. Esta última fase es desarrollada por el grupo de estudio de LTE (evolución a largo plazo). Podrá alcanzar estándares de hasta 100 Mbps en el enlace descendente.

En la actualidad la fase 1 esta desplegado comercialmente, encontrándose en despliegue la fase 2. La fase 3 se encuentra en fase piloto.

HSDPA es ya una tecnología implantada, por lo que posee un grado de madurez mayor que WiMAX. Las ventajas que WiMAX Móvil puede aportar quedan compensadas por el coste asociado a la obtención de nuevas licencias. Son necesarios, adicionalmente, estudios que certifiquen que WiMAX Móvil y HSDPA pueden compartir espectro e incluso localizaciones sin causarse interferencias.

A pesar de las ventajas tecnológicas de WiMAX, los grandes operadores han apostado decididamente por HSDPA, que esta evolucionando para alcanzar prestaciones similares a WiMAX Móvil

Sin embargo, WiMAX Móvil se plantea como un complemento en situaciones no fácilmente cubiertas por otras alternativas, como hotspots de gran tamaño en entornos urbanos para aplicaciones específicas o en entornos rurales.

2.2.2 WiMAX Móvil frente Wi-Fi

Estrictamente hablando, Wi-Fi y WiMAX no son protocolos sobre los que pueda hablarse en términos de enfrentamiento, pues han sido diseñados con propósitos distintos.

Wi-Fi.

Pertenece a la familia de protocolos IEEE 802.11. Su objetivo es proporcionar técnicas inalámbricas de acceso a Redes de Área Local (LAN). Cubre áreas pequeñas (domicilios y oficinas pequeñas), con líneas de visión directa de decenas de metros. Mediante el empleo de varios puntos de acceso Wi-Fi se pueden cubrir áreas más grandes, como edificios o campus.

Los dos protocolos más extendidos son IEEE 802.11b y 802.11g, que emplean la banda de 2.4 GHz, que es una banda sin licencia y, por tanto, sujeta a interferencias. Pese a sus limitaciones, algunos ISP (Proveedores de Servicios de Internet) emplean esta tecnología en la "última milla", con lo que la capacidad prestada al usuario final es limitada.

Wi-Fi ha incorporado varias mejoras, como VLAN (Red de Área Local Virtual), seguridad ampliada, soporte elemental para servicios de voz con calidad de servicio.

El último estándar Wi-Fi es el 802.11n, también denominado Wi-Fi N, que incluye el empleo de técnicas MIMO y el uso simultaneo de las bandas de 2.4 y 5 GHz.

WiMAX

Como ya se ha comentado, WiMAX es un estándar específicamente diseñado para proveer una solución inalámbrica para la "última milla" una Red de Área Metropolitana (MAN). Esto permite proveer a los usuarios acceso directo mediante el empleo de hotspots WiMAX (Móvil) en grandes áreas como campus universitarios o empresariales.

Adicionalmente, permite a los operadores proveer servicios de banda ancha sus suscriptores mediante WiMAX (Fijo) sin necesidad de cablear desde la centralita local y el domicilio del usuario. En este caso se emplearían un CPE. En el domicilio del usuario se podría empelar Wi-Fi para dar acceso a los terminales finales.

WiMAX ha sido diseñado para operar en bandas de frecuencia con licencia, sometidas a regulación y requiere una planificación de despliegue similar al de las redes de telefonía móvil GSM Por tanto, Wi-Fi y WiMAX no son estándares enfrentados sino complementarios, como se muestra en la figura 7, por lo que no será extraño la proliferación de dispositivos que soporte ambos estándares.

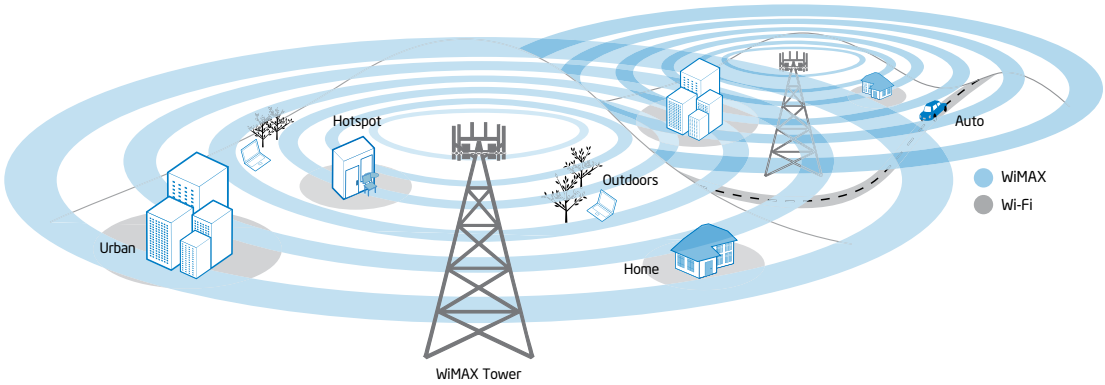

Figura 7. Complementariedad de Wi-Fi y WiMAX. Fuente: Intel.

2.3 Despliegues WiMAX.

Desde el momento de la aprobación del estándar IEEE 802.16d-2004 (WiMAX Fijo) los operadores de telecomunicaciones mundiales mostraron un creciente interés, reforzado tras la posterior aprobación de IEEE 802.16e-2005 (WiMAX Móvil). En la figura 8 se muestran los principales despliegues a nivel mundial (la figura muestra solo las sedes principales de los despliegues)

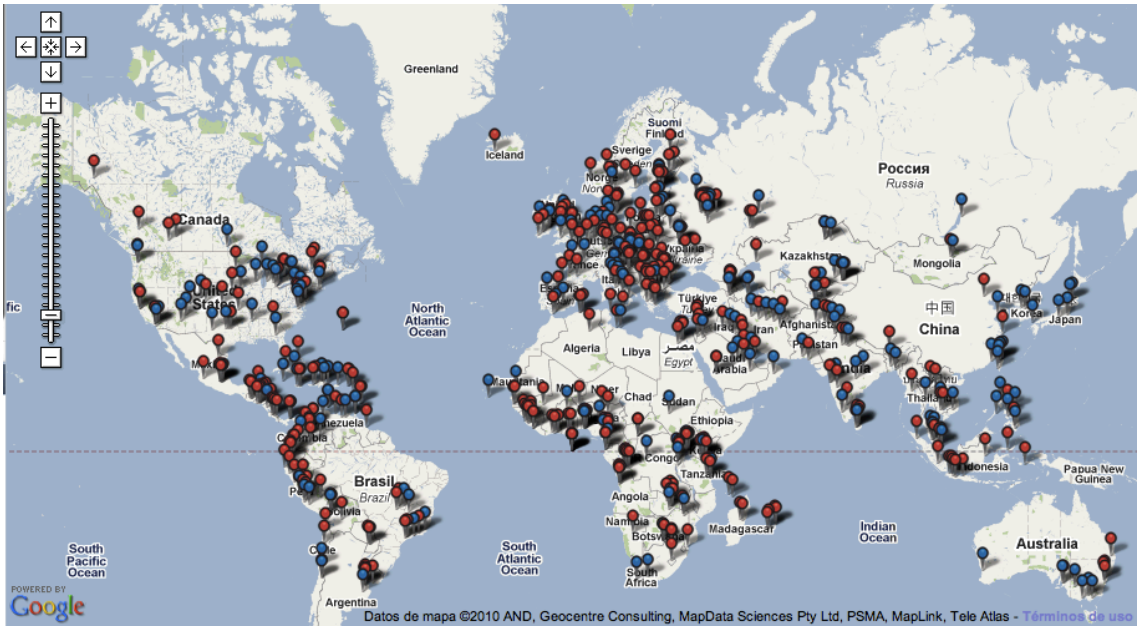

Figura 8. Despliegues mundiales de WiMAX Fijo (Rojo) y Móvil (Azul). Fuente: WiMAX Forum

Igualmente, en España el interés de las operadoras de telecomunicaciones ha sido considerable. Los principales despliegues de WiMAX en España, de acuerdo con la herramienta de seguimiento de WiMAX Forum [5], se muestran en la figura 9.

Figura 9. Despliegues de WiMAX Fijo (Rojo) y Móvil (Azul) en España. Fuente: WiMAX Forum. Nota: Se muestra un único pin por despliegue, independientemente del número de localizaciones del mismo.

A continuación se muestra una tabla con las primeros despliegues y pruebas piloto, muchos de los cuales fueron posibles por el impulso de las instituciones públicas españolas, que se han mostrado interesadas en la tecnología dada su capacidad de prestar servicio en zonas rurales o deficientemente comunicadas

Año	Ubicación	Operador	Institución	Observaciones
2003	Cataluña	Iberbanda	Centre de Telecomunications i Tecnologies de la Informació (CTTI) de la Generalitat de Catalunya	En concurso público, la Generalitat consideró la solución WiMAX de Iberbanda como la mejor para proporcionar cobertura de banda ancha en zonas rurales de Lleida y Tarragona.
2005	Andalucía	Iberbanda	-	Fue la primera experiencia comercial de WiMAX con tecnología Intel en Europa, prestando servicio a 50 clientes en localidades de la provincia de Almería.
	Valencia	-	(ITACA)	El Instituto de Aplicaciones de las Tecnologías de la Información y de las Comunicaciones Avanzadas, ITACA, instaló una antena de emisión WiMAX, que alcanzó una cobertura de 20 kilómetros alrededor con velocidades medias de 10 Mbps.
	Baleares	Telefónica	-	Primera experiencia a nivel mundial de “navegación simultanea por el mar e internet”. Se situó una estación base en Sierra de Na Burguesa, Palma de Mallorca, que cubría la Bahía de Palma. Se equipó un velero con un CPE (equipo receptor) en la parte superior de uno de sus mástiles. Se alanzaron velocidades sostenidas de 2 Mbps.
	Castilla y León	Iberbanda	Junta de Castilla y León	Iberbanda fue seleccionada para llevar WiMAX a zonas rurales de la comunidad, fruto del programa de Banda Ancha 2005 – 2007, que buscaba proveer de las mismas oportunidades de acceso a todos los municipios de la región.
	Galicia	Galileo R	-	Prestación de servicios basados en WiMAX
2006	Navarra	Iberbanda	Gobierno de Navarra	El Gobierno de Navarra encargó a Iberbanda el despliegue de WiMAX en 855 municipios rurales. El objetivo era dar acceso de banda ancha a 75.000 ciudadanos con velocidades de entre 512 kbps y 4 Mbps
	Aragón	Telefónica y Embou	Gobierno de Aragón	Lanzamiento de prueba piloto en las comarcas de Sobrarbe y Ribagorza. 19 postes daban cobertura al 90% de la población de 27 municipios de estas comarcas.
	País Vasco	Euskaltel		Euskaltel comenzó a ofrecer conexiones WiMAX de forma comercial

Como se puede apreciar en la tabla anterior, la compañía Iberbanda ha resultado adjudicataria de diversos concursos y programas impulsados por distintas Administraciones Públicas, con el objeto de incentivar la extensión de la banda ancha al ámbito rural.

Fruto de esta actividad, Iberbanda ha alcanzado el número de 1.000 estaciones base desplegadas en España (17/02/2010). Entre los últimos despliegues destacan:

- Sur de Lugo (finales de 2010): La solución ofertada por Iberbanda emplea la tecnología sin hilos Wimax Móvil de banda licenciada, que presta un servicio mínimo exigido de 2 Mbps de velocidad de bajada y 512 Kbps de subida con un 20% de caudal garantizado el 95% del tiempo. Cubre más de 2.000 núcleos de 27 concellos del sur de Lugo, al amparo del Plan de Banda Larga 2010-2013 de la Xunta de Galicia.

- Navarra (finales de 2010): Ampliación del acuerdo firmado en 2005. Se han instalado 90 estaciones base, que prestan servicio a 4.600. Se espera llegar a 6.000. La red cubre asimismo las necesidades generadas por la propia administración.

- Orense: Instalación de 40 estaciones base WIMAX-e (IEEE 802-16e), 43 radioenlaces punto a punto y 15 emplazamientos de nueva construcción. Se da cobertura a 1.540 núcleos en 51 concellos, con una población total de más de 110.000 personas.

3. CARACTERIZACIÓN DE LA PLATAFORMA EMPLEADA

La plataforma utilizada en la realización del presente proyecto para el prototipado del sistema real (sistema DSP), consta de un elemento hardware y de un elemento software, plenamente compatibles:

3.1 Hardware: VHS-ADC/DAC Virtex-4.

Placas de adquisición/generación de datos analógicos que cuenta con una FPGA de alta capacidad integrada. La placa con denominación VHS-ADC (Conversor Analógico a Digital) adquiere datos analógicos, siendo empleada para implementar el receptor, mientras que la placa VHS-DAC (Conversor Digital a Analógico) genera datos analógicos, dando soporte físico al emisor.

3.2 Software: System Generator for DSP de Xilinx, conjuntamente con MATLAB® y Simulink®

System Generator for DSP de Xilinx aporta una serie de bloques propios que pueden ser empleados en el entorno Simulink®, para realizar simulaciones. Estos bloques pueden ser posteriormente sintetizados, mediante la generación del código VHDL (Lenguaje de Descripción de Soportes Físicos para Circuitos Integrados Muy Rápidos) correspondiente, sin pérdida de prestaciones en comparación con diseños realizados directamente en VHDL

A continuación se describen con más detalle las placas VHS-ADC/DAC Virtex-4 y los bloques Xilinx empleados en este proyecto.

Los anexos D y E aportan información adicional sobre MATLAB® y Simulink® .

3.3 VHS-ADC Virtex-4 de Lyrtech

Las placas VHS-ADC y VHS-DAC Virtex-4 de Lyrtech son dos placas compatibles destinadas a la adquisición de datos análógicos (VHS-ADC) y la converción analógica/digital (VHS-DAC). Estas placas fueron empleadas para la realización del proyecto de elaboración del sistema completo emisor-receptor. Sin embargo, dado que ambas plataformas permiten realizar las tareas de simulación de forma indistinta, se empleó la plataforma VHS-ADC durante la mayor parte del proyecto.

Estas placas (ver figura 10) aportan Integración completa con MATLAB®, Simulink® y *System Generator for DSP*, lo cual permite un alto nivel de abstracción en el diseño de modelos y su compilación automática en la FPGA, generándose el código VHDL correspondiente.

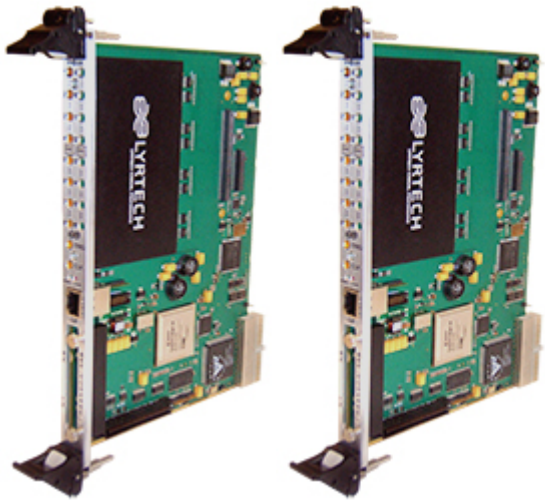

Figura 10. VHS-DAC (derecha) y VHS-ADC (izquierda) Virtex-4 de Lyrtech

Estas placas comparten la mayor parte de carácterísticas:

- Conexión al PC mediante Compact PCI (Interconexión de Componentes Periféricos Compactos) del tipo 6U, lo cual le permite instalarse en un chasis cPCI estándar.
- Posibilidad de utilizar la placa sin una conexión cPCI mediante el empleo de la memoria flash que posee integrada para la FPGA y el puerto externo I2C/JTAG.
- FPGA Virtex-4 de bajo consumo integrada en la placa con 256 GMACS (Gigaoperaciones Acumuladas de Multiplicación Por Segundo) y 152.000 células lógicas.

- Memoria SDRAM (Memoria Dinámica de Acceso Aleatorio Síncrono) integrada de 128 MB (Megabytes) de capacidad para almacenar y reproducir datos mediante una herramienta software incluida.
- Puertos digitales RapidCHANNEL (uno de transmisión y otro de recepción) que alcanzan hasta 8 GBps (Gigabyte Por Segundo) en full-dúplex.
- Conector de expansión que permite agregar 8 canales adiciones de entrada/salida (ADC/DAC) y varios gigabytes de memoria DDR2 SDRAM (SDRAM de Tasa Doble de Transferencia de Datos).
- Ganancia es independiente para cada canal y programable por software, siendo la entrada por defecto del tipo 50 Ω MMCX (Coaxial Micro Miniaturizado).
- Control de los parámetros de la placa mediante una herramienta software.
- Kit software de desarrollo para la placa basado en los lenguajes de programación VHDL y C

Las placas se diferencian en los conversores que montan por defecto:

- VHS-ADC

  8 ADC (Conversores Analógico a Digital) integrados, con una capacidad máxima, por cada uno, de 105 Megamuestras por segundo y una resolución de 14 bits por muestra

- VHS-DAC

  8 DAC (Conversores Digital a Analógico) integrados, con una capacidad máxima, por cada uno, de 480 Megamuestras por segundo, empleando tecnología de interpolación (4x), y una resolución de 14 bits por muestra

A continuación se muestran los esquemas de bloque de la placa VHS-ADC (figura 11) y VHS-DAC (figura 12) Virtex-4 de Lyrtech.

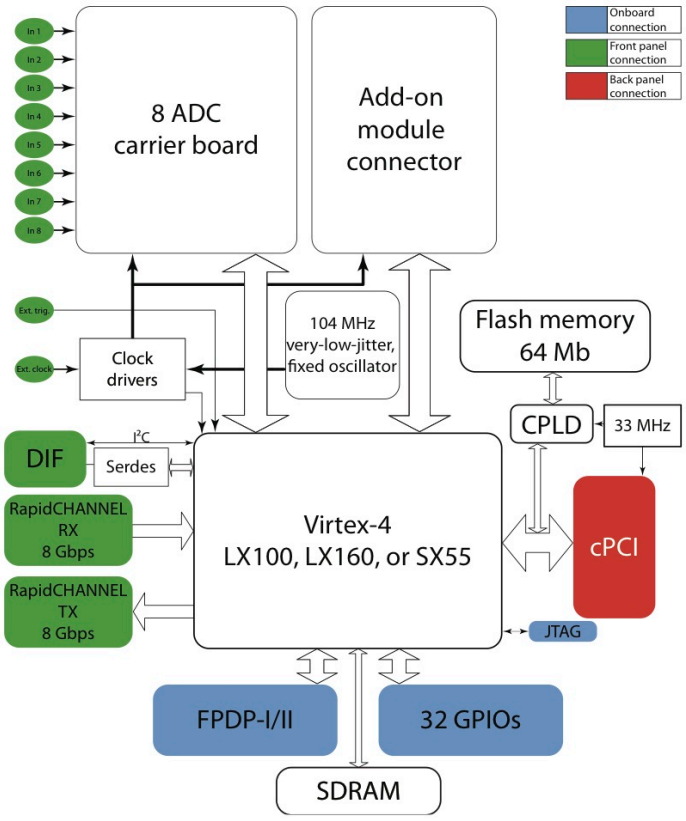

Figura 11. Diagrama de Bloques VHS-ADC Virtex 4. Fuente: Lyrtech

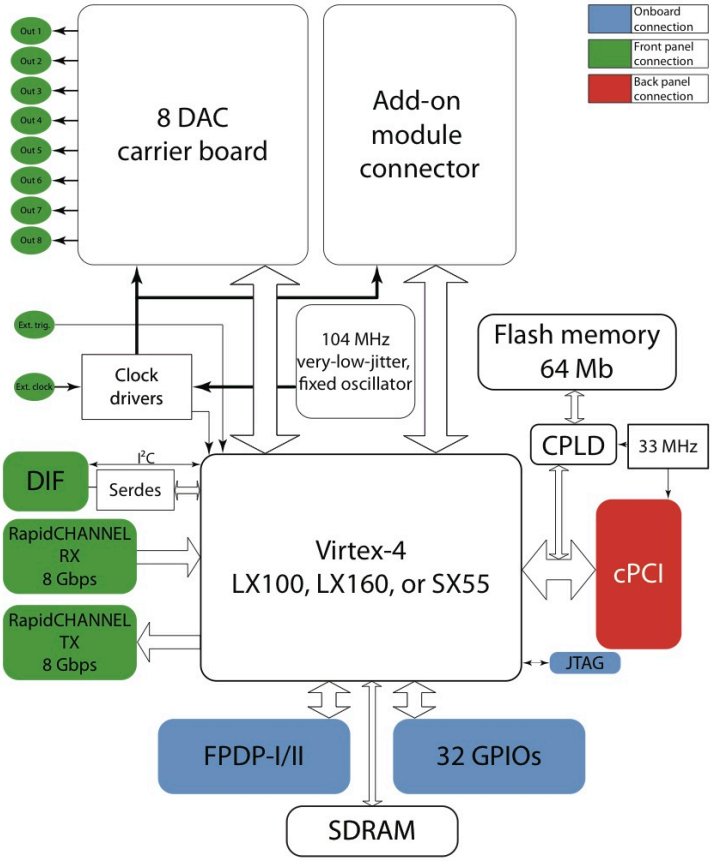

Figura 12. Diagrama de Bloques VHS-DAC Virtex 4. Fuente: Lyrtech

Las **aplicaciones** más importantes en las cuales destacan la *VHS-ADC y la VHS-DAC Virtex-4* son

Tabla 2. Principales Aplicaciones de VHS-ADC y VHS-DAC Virtex-4. Fuente: Lyrtech.

Aplicación	VHS-ADC Virtex-4	VHS-DAC Virtex-4
Estaciones Base Avanzadas	●	●
Antenas inteligentes, sistemas a frecuencia intermedia multicanal y conformadores de haz.	●	●
Aplicaciones inalámbricas (routers, OFDM, diversidad de antena, WiFi, WiMAX)	●	●
Codificación espacio-tiempo MIMO	●	●
Equipos SDR (Radio Definida por Software)	●	●
Generación y reproducción multicanal de señales a ata velocidad.		●
Geolocalización (basada en TDOA, DOA y ADOA)	●	
Radar, radar basado en una agrupación de antenas controladas por fase (*phased-array radar*)		●
Comunicaciones por satélite		●
Sistemas de medida y testeo a alta velocidad	●	
Medicina, tomografía, ultrasonidos y otras aplicaciones		●
Simulación de canales	●	●

Nota 1: ● = SI.

Tanto el *VHS-ADC* como el *VHS-DAC Virtex-4* se suministran con un kit de desarrollo de la placa (Board Software Development kit). El BDSK permite acceder la FPGA a través de los proyectos *ISE Foundation* y viene acompañada de documentación básica de la FPGA. Adicionalmente, el BDSK incluye una API y programas que permiten la comunicación con la FPGA. Finalmente, el BDSK incluye un set completo de ejemplos funcionales de hardware que demuestran como usar las entradas/salidas y las interfaces.

Opcionalmente, se puede adquirir el kit de diseño basado en modelos (Model-based design kit, MBDK), que permite generar código para la FPGA desde MATLAB y Simulink a través de *System Generator for DSP*. Con Simulink, es posible modificar parámetros en el acto y realizar co-simulaciones sobre la propia placa. Finalmente, el BDSK incluye un set completo de ejemplos funcionales de hardware que demuestran como usar las entradas/salidas y las interfaces en un entorno de diseño basado en modelos.

También opcionalmente, es posible adquirir *Diamond*, de 3L, que mejora las herramientas de procesador único con un modelo multiprocesador simple y probado que ofrece un nivel de abstracción que conduce a sistemas eficientes, coherentes y fiables.

System Generator for DSP

System Generator for DSP es un producto sofware desarollado y comercializado por Xilinx. System Generator for DSP permite al desarrollador tener una visión de alto nivel de su sistema DSP a través del entorno Simulink®, en el que se integra. Desde ese entorno es posible realizar todo el trabajo de simulación. Esta herramienta permite, asimismo, la posterior generación del código VHDL correspondiente y su implementación en FPGA.

Las principales características de *System Generator for DSP* de Xilinx ® son las siguientes:

- **Modelado DSP.**

  Generación y simulación de sistemas DSP de alto rendimiento, en Simulink®, utilizando el conjunto de bloques Xilinx, que incluye funciones para el procesado de señal como:

  - Filtros FIR (Respuesta Finita al Impulso)
  - FFT (Transformada Rápida de Fourier)
  - Corrección de errores: decodificador Viterbi, decodificador/codificador Reed Solomon)
  - Funciones aritméticas
  - Memorias: FIFO (Primero en Entrar Primera en Salir), RAM (Memoria de Acceso Aleatorio), ROM (Memoria de Sólo Lectura)
  - Lógica Digital

- **Generación automática de código en VHDL o Verilog desde Simulink®**.

  Desde Simulink® se genera de forma automática el código VHDL o Verilog correspondiente al modelo diseñado, sin perder prestaciones con respecto a un código VHDL generado directamente.

  También permite la incorporación de código HDL a medida, a través de su flujo de importación de HDL.

- **Co-simulación Hardware**.

  Opción de generación de código que permite validar el modelo integrando el hardware en funcionamiento con el entorno de Simulink®, que captura y genera los datos desde y para el Hardware.

System Generator for DSP de Xilinx es compatible con las siguientes familias de dispositivos:

- Virtex-II.
- Virtex-II Pro.
- Virtex-E.
- Virtex-4 FX, LX, SX.
- Virtex-5 LX, LXT, SXT, FXT.
- Spartan-II, IIE.
- Spartan-3A, AN.
- Spartan-3A DSP.
- Spartan-3, 3E.

3.4 Descripción de los bloques empleados

En este apartado se abordará en detalle las características de los pricipales bloques empleados en el emisor diseñado y que pertenecen al Xilinx Blockset de Xilinx System Generator for DSP. Para cada uno de ellos se ofrecerá una ficha que describirá sus características y opciones de funcionamiento, junto con un modelo demostrador.

3.4.1 Bloque Lyrtech VHS-ADAC Board Configuration

Descripción

El bloque Lyrtech VHS-ADAC Board Configuration (figura 13) pemite configurar el hardware sobre el cual será sintetizado el modelo diseñado mediante System Generator for DSP.

Figura

Figura 13. Bloque Lyrtech VHS-ADAC Board Configuration

Configuración Básica

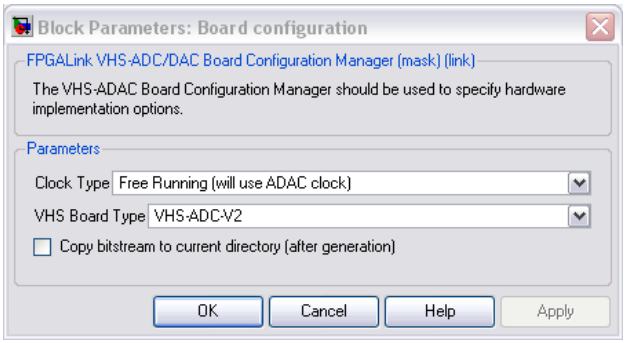

Figura 14. Ventana de Configuración básica del bloque Lyrtech VHS-ADAC Board Configuration

Parámetros

1 ***Clock Type*** - Tipo de Reloj

Reloj que empleará el diseño basado en *System Generator for DSP.*

Puede ser: **Single Step** (empleado para la co-simulación o simulación sobre la FPGA) o **Free Running** (utilizado para el resto de operaciones con el modelo).

2 ***VHS Board Type*** - Tipo de Placa VHS

Placa empleada por el modelo basado en *System Generator for DSP*

Puede ser: **VHS-ADC-V2**, **VHS-DAC-V2** o **VHS-ADC-V4**

3 ***Copy bitstream to current directory*** - Copiar flujo de bits al directorio actual

Indica si se desea copiar el flujo de bits de la FPGA al directorio de trabajo actual de MATLAB®.

Observaciones

- Este bloque solo es necesario si se va a involucrar al Hardware. No es necesario para tareas exclusiva de simulación.
- Este bloque debe acompañarse del Xilinx System Generator, en el que se especifica el periodo del reloj de la FPGA (como se explicará en la ficha correspondiente)
- Este bloque lo suministra directamente el fabricante del Hardware, Lyrtech.

3.4.2 Bloque Xilinx System Generator

Descripción

El bloque *Xilinx System Generator* (figura15) proporciona el control del sistema y de los parámetros de simulación a nivel general del modelo diseñado. Además, mediante este bloque se puede indicar cómo se realizará la generación de código, así como invocar al propio generador.

Figura

Figura 15. Bloque Xilinx System Generator

Configuración Básica

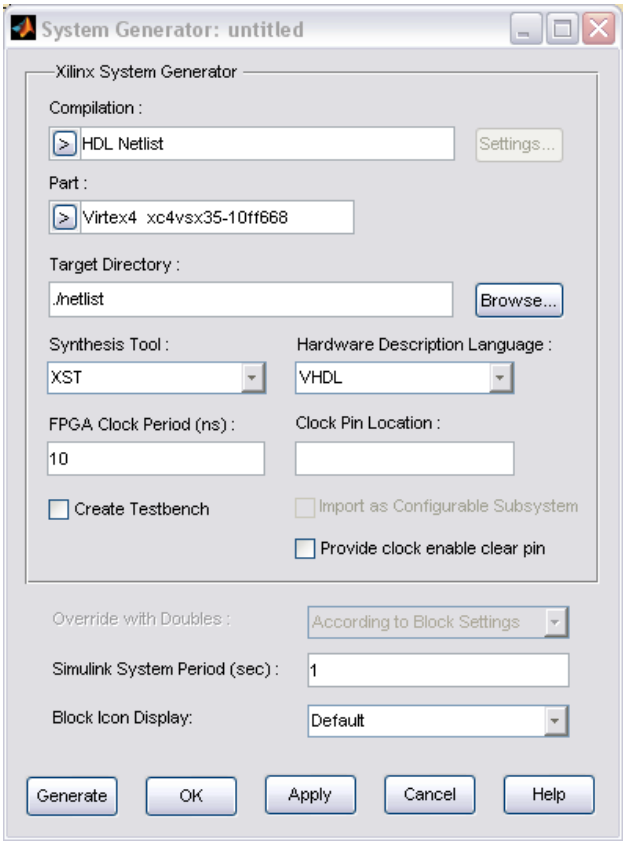

Figura 16. Ventana de Configuración básica del bloque Xilinx System Generator

Parámetros

1 ***Compilation*** - Compilación

Tipo de compilación resultante tras invocar al generador de código.

Puede ser: **HDL Netlist**, **NGC Netlist**, **Bitstream**, **EDK Export Tool**, **Hardware Co-Simulation**, **Lyrtech** o **Time Analysis**.

2	***Part*** - componente Tipo de FPGA empleada *Puede ser*: **Spartan2, Spartan2E, Spartan3, Spartan3E, Virtex, VirtexE, Virtex2, Virtex2P** o **Virtex4.**
3	***Target directory*** - Directorio de destino Directorio en el cual *System Generator* escribirá los resultados de la compilación
4	***Synthesis Tool*** - Herramienta de Síntesis Herramienta empleada para sintetizar el modelo diseñado Puede ser: **Synplify**, **Synplify Pro** (ambas de Synplicity) o **XST** (de Xilinx).
5	***Hardware Description Language*** - Lenguaje de Descripción de Hardware Lenguaje HDL utilizado para compilar el diseño Puede ser: **VHDL** o **Verilog**
6	***FPGA Clock Period*** - Periodo de Reloj Valor, en nanosegundos, del período del reloj del hardware. Este valor es enviado a las herramientas de implementación de Xilinx mediante un fichero de restricciones y es usado como el periodo global, empleado el resto de elementos múltiplos enteros del mismo. No tiene porque ser entero. De
7	***Clock Pin Location -*** Localización del pin del reloj Posición del pin del reloj hardware. Esta información es enviada a las herramientas de implementación de Xilinx mediante un fichero de restricciones
8	***Create Testbench*** - Crear Banco de Pruebas Hace que *System Generator* cree un banco de pruebas en HDL, comparando los resultados de la simulación en Simulink® con los obtenidos mediante la versión compilada del modelo. Este parámetro se activa mediante casilla de verificación.
9	***Import as Configurable Subsystem*** - Importar como un subsistema configurable. Señala a System Generator que se desea crear un subsistema configurable. Este parámetro se activa mediante casilla de verificación.
10	***Provide clock enable clear pin*** - Proporcionar pin de borrado de habilitado del reloj Con esta opción activa, System Generator crea un puerto accesible por el

desarrollador que permite manejar la lógica de funcionamiento del reloj.

Este parámetro se activa mediante casilla de verificación.

11 ***Override with Doubles*** - Anular con precisión doble

Esta opción indica que todos los cálculos en el ámbito del bloque deben ser realizados con aritmética de precisión doble.

12 ***Simulink System Period -*** Período del Sistema Simulink

Este periodo es el Máximo Común Divisor de todos los períodos de muestro implicados por todos los bloques del modelo. Si se especifica un divisor menor, en el momento de la simulación el sistema generará un mensaje de advertencia, recomendando el Máximo Común Divisor. Dentro del modelo, los periodos relativos sobre este parámetro lo son de la misma manera sobre el periodo de la placa (ver observaciones).

13 ***Block Icon Display*** - Visualización en el Icono de Bloque

Mediante esta campo se selecciona el tipo de información que se quiere ver en el icono de cada bloque. Esta información se actualiza tras cada compilación del modelo.

Las opciones don: **vista por defecto**, **tasas de muestreo** (periodos de muestreo normalizados al valor de Simulink System Period (parámetro 12), **nombres en HDL de los puertos**, **tipos de datos a la entrada** y **tipos de datos a la salida**.

Observaciones

- La presencia de este bloque es necesaria en cualquier modelo en el que aparezcan un bloque perteneciente al Xilinx Blockset, independientemente de que solo se quiera simular.
- Debe tenerse en cuenta que si se desea proceder completamente con el ciclo descrito en los objetivos del presente documento, lo que requiere completar la síntesis del modelo tras su diseño y simulación, la frecuencia de reloj que se debe fijar se corresponde con la frecuencia de la placa que se va a usar.

  En este sentido, cabe destacar que, tal como muestra figura 16, en el caso del modelo presentado se ha escogido un tiempo de reloj de 10 ns. En el caso del presente proyecto, se empleo la placa que opera como ADC, a 105 megamuestras por segundo, esto es, 9,52 ns.

  El hecho de que el sistema mantenga las dos variables por separado, dota al desarrollador de un mayor nivel de abstracción y permite que los modelos sean

implementables en placas distintas, mediante el cambio de los parámetros de la placa, dado que el sistema mantiene la relación de frecuencias sobre todo el modelo.

3.4.3 Bloque Xilinx WaveScope

Descripción

El bloque Xilinx WaveScope (figura 17) permite observar, tras la simulación del modelo, los valores de las señales en las líneas que conectan los bloques, en modo cronograma.

Este bloque permite ver simultáneamente varias señales y configurar esta visualización de la manera más adecuada en cada caso.

Figura

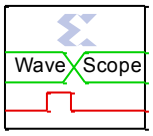

Figura 17. Bloque Xilinx WaveScope

Configuración Básica

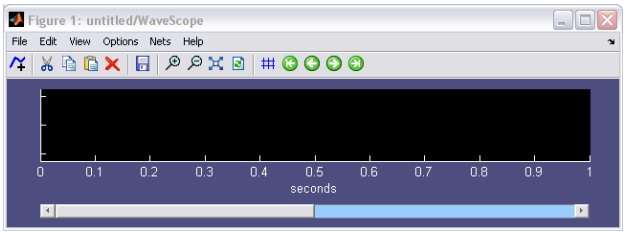

Figura 18. *: Ventana de Visualización y Configuración básica del bloque Xilinx WaveScope*

Configuración

Este bloque se trata de un bloque de visualización de señales, por lo que no consta de una ventana de configuración como la del resto de bloques. La configuración se realiza desde la misma ventana mediante la barra superior.

Las funcionalidades más relevantes de esta ventana incluyen:

- Selección de señales a visualizar.

  Todas las señales del modelo son accesibles de esa ventana. La señal deseada se agrega desde un menú desplegable. También es posible agregar una señal desde el propio modelo, seleccionando la línea de datos por la que fluye y eligiendo la opción correspondiente en el menú contextual.

- Configuración de la presentación de la señal.

  Mediante el botón secundario del ratón se puede elegir el formato, base y color de cada señal agregada, además de asignarle un nombre.

- Visualización de señales.

El tiempo de respuesta de este bloque, se degrada rápidamente en función del volumen de información (número, tamaño y duración de las señales). Esta degradación se traduce en mayores tiempos de respuesta cuando se solicita un desplazamiento o un zoom. Para mantener el nivel de prestaciones se ofrece la posibilidad de seleccionar los valores que se grabarán y mostrarán que, por defeco, son todos los de la simulación.

Observaciones

- Siempre que exista una señal aparecerá la señal de reloj como base del cronograma. Necesariamente, dicha señal debe ser la más rápida del sistema.

3.4.4 Bloque Xilinx Counter

Descripción

El bloque *Xilinx Counter* (figura 19) consiste en un bloque que genera un flujo de bits según una cuenta definida.

Figura

Figura 19. Bloque Xilinx Counter

Puertos de entrada y salida

Salida

S-1	***Output*** - Salida Datos generados según la cuenta definida.

Configuración Básica

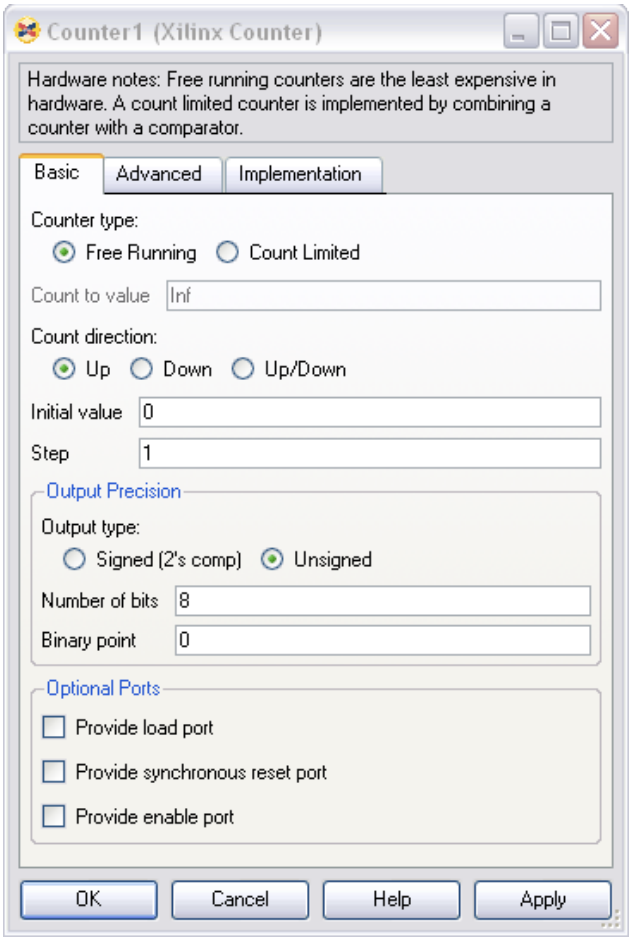

Figura 20. Ventana de Configuración básica del bloque Xilinx Counter

Parámetros	
1	***Counter Type*** – Tipo de Conteo
	Modo en que se realiza la cuenta. Puede ser en modo `Free Running` (cuenta libre), en cuyo caso el contador agota la mantisa de bits definida y vuelve a empezar o `Count limited` (cuenta limitada), donde el contador llega al valor definido y vuelve a empezar.
2	***Count direction*** – Dirección de conteo
	Define el sentido de la cuenta, ascendente, descendente o ascendente/descendente. En este último caso, el conteo comienza ascendente y cuando llega al límite se vuelve descendente.
3	***Initilal Value*** – Valor Inicial
	Establece el punto dese el que se empieza a contar
4	***Step*** – Paso
	Establece el incremento unitario del contador.
5	***Output Precision*** - Precisión de Salida
	Establece el formato de la salida. Se puede establecer el criterio de signo (con o sin él) el número de bits y la posición del punto binario.
6	***Optional Ports*** - Puertos Opcionales
	Permite habilitar puertos opcionales a la entrada: ▪ ***Load Port*** - Puerto de Carga: Activado, el bloque funciona en modo de cuenta libre con un puerto explícito de carga y entrada de datos. ▪ ***Synchronous reset port*** – Puerto de reset síncrono: Permite resetear la cuenta. ▪ ***Enable Port*** – Puerto de habilitación: Habilita o deshabilita al contador.

Ejemplo de Funcionamiento

Modelo

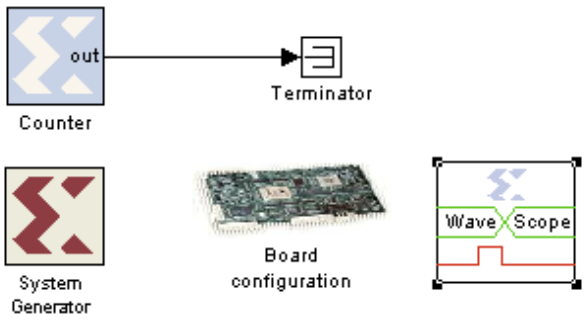

Figura 21. Modelo de ejemplo de funcionamiento del bloque Xilinx Counter

Descripción

Este modelo consta de:

- Un contador, configurado en modo de cuenta libre ascendente, 3 bits y paso de dos, por lo que solo contará los pares.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Este sistema retrasa tres períodos de muestra los datos de entrada, tal y como se verifica en las figuras 22 y 23

Configuración

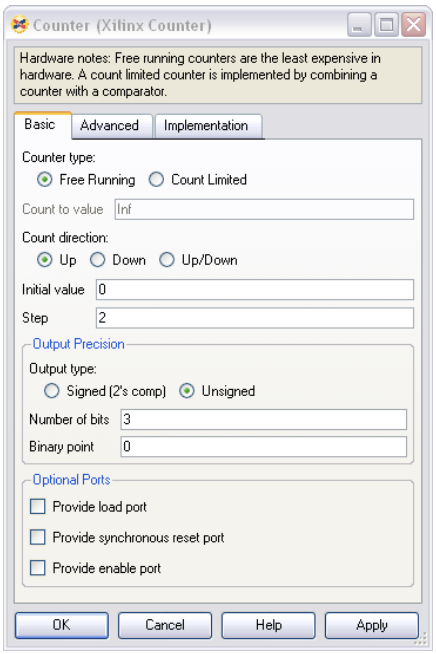

Figura 22. Configuración para el ejemplo funcionamiento del bloque Xilinx Counter

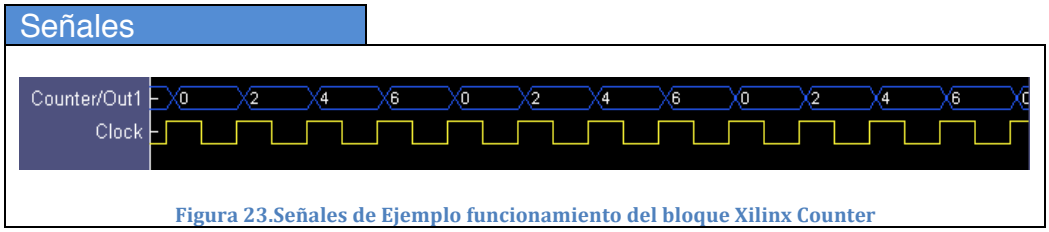

Figura 23.Señales de Ejemplo funcionamiento del bloque Xilinx Counter

3.4.5 Bloque Xilinx Convert

Descripción

El bloque *Xilinx Convert* (figura 24) convierte el flujo de datos al formato indicado

Figura

Figura 24. Bloque Xilinx Convert

Puertos de entrada y salida

Entrada

E-1	***Input*** - Entrada Datos que se quieren convertir

Salida

S-1	***Output*** - Salida Datos convertidos al formato indicado

Configuración Básica

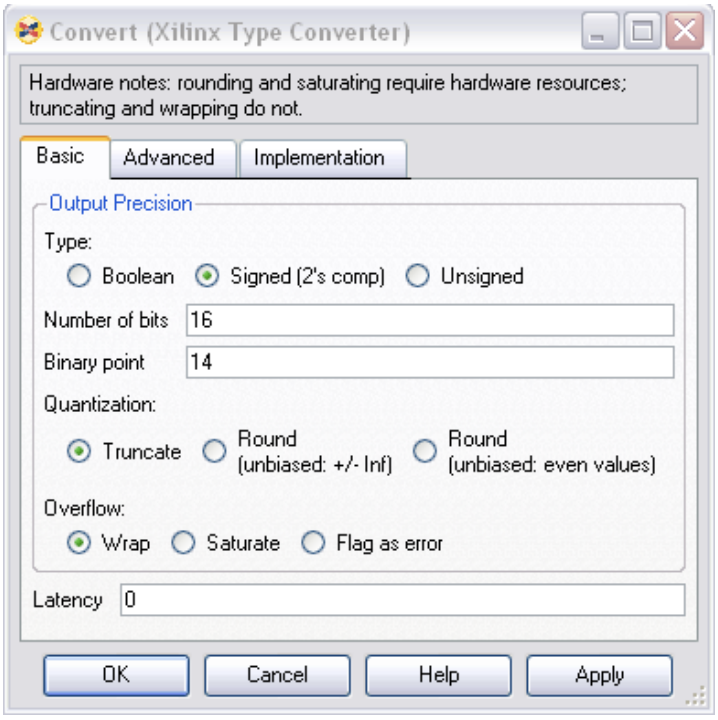

Figura 25. Ventana de Configuración básica del bloque Xilinx Convert

Parámetros

1	***Type*** - Tipo
	Número de períodos de muestreo que se desea retardar los datos a la entrada.
2	***Number of bits*** – Número de bits
	Establece el número de bits del flujo de datos de salida
3	***Binary Point*** – Punto Binario
	Establece la posición del punto binario en el flujo de datos de salida
4	***Quantization*** – Cuantificación
	Fija el modo de cuantifiación. Se puede elegir entre truncamiento y dos modos de redondeo.
5	***Overflow*** - Desbordamiento
	Fija el comportamiento ante un desbordamiento. Puede dar la vuelta a la mantisa, saturar la mantisa o activar un flag de error.
6	***Latency*** - Latencia
	Fija un retardo (0 por defecto) entre la entrada y la salida

Ejemplo de Funcionamiento

Modelo

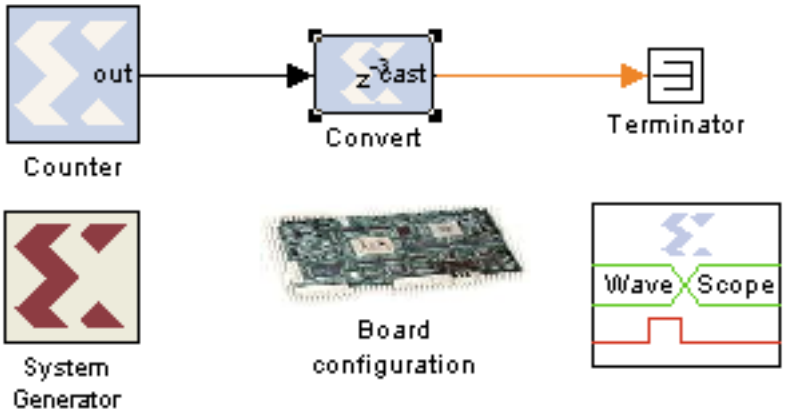

Figura 26. Modelo de ejemplo de funcionamiento del bloque Xilinx Convert

Descripción

Este modelo consta de:

- Una fuente de datos, papel desempeñado por bloque Xilinx Counter configurado como contador libre de 3 bits sin signo.
- Un bloque Xilinx Convert, configurado para conformar una salida con de 4 bits enteros y dos decimales, con un retardo de tres períodos de muestreo.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Este sistema convierte un flujo de tres bits enteros en uno de cuatro bits enteros y dos binarios y retrasa tres períodos de muestra tal y como se verifica en las figuras 27 y 28

Configuración

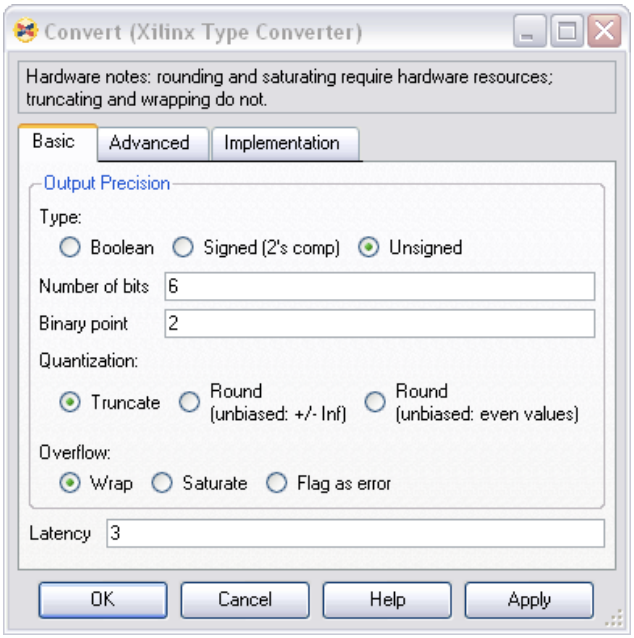

Figura 27. Configuración para el ejemplo funcionamiento del bloque Xilinx Convert.

Señales

Figura 28.Señales de Ejemplo funcionamiento del bloque Xilinx Convert.

3.4.6 Bloque Xilinx Delay

Descripción

El bloque *Xilinx Delay* (figura 29) consiste en un bloque de retardo que agregar la latencia deseada a la línea de datos en la que se conecte..

Figura

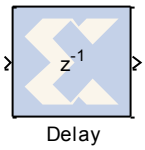

Figura 29. Bloque Xilinx Delay

Puertos de entrada y salida

Entrada

E-1	***Input*** - Entrada Datos que se quieren retardar

Salida

S-1	***Output*** - Salida Datos retardados

Configuración Básica

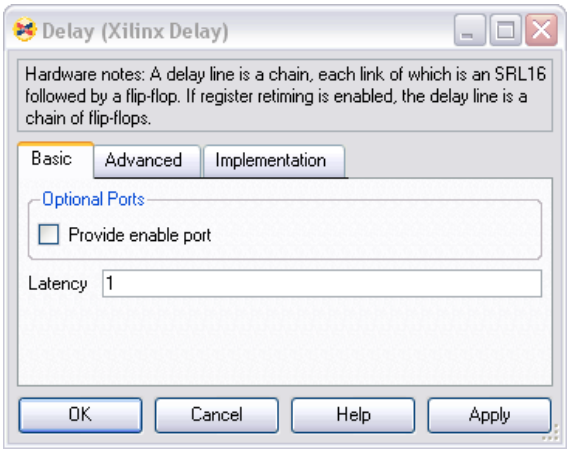

Figura 30. Ventana de Configuración básica del bloque Xilinx Delay

Parámetros

1	***Latency*** - Latencia Número de períodos de muestreo que se desea retardar los datos a la entrada.
2	***Provide Enable Port*** - Proporcionar Puerto de Habilitación De activarse esta opción, el bloque mostrará un puerto adicional de habilitación. La señal que controle este puerto ha de ser de tipo booleano.

Ejemplo de Funcionamiento

Modelo

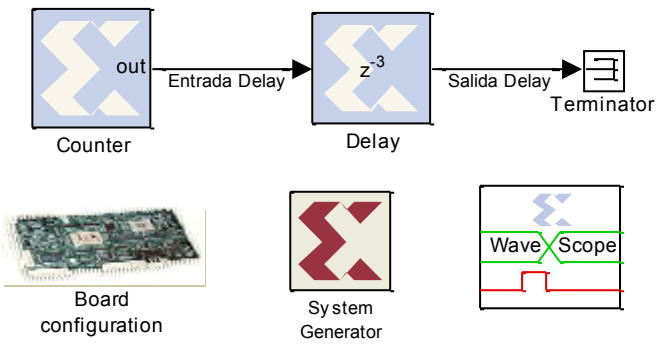

Figura 31. Modelo de ejemplo de funcionamiento del bloque Xilinx Delay.

Descripción

Este modelo consta de:

- Una fuente de datos, papel desempeñado por bloque Xilinx Counter configurado como contador libre de 4 bits sin signo.
- Un bloque Xilinx Delay con un retardo de tres períodos de muestreo.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Este sistema retrasa tres períodos de muestra los datos de entrada, tal y como se verifica en las figuras 32 y 33

Configuración

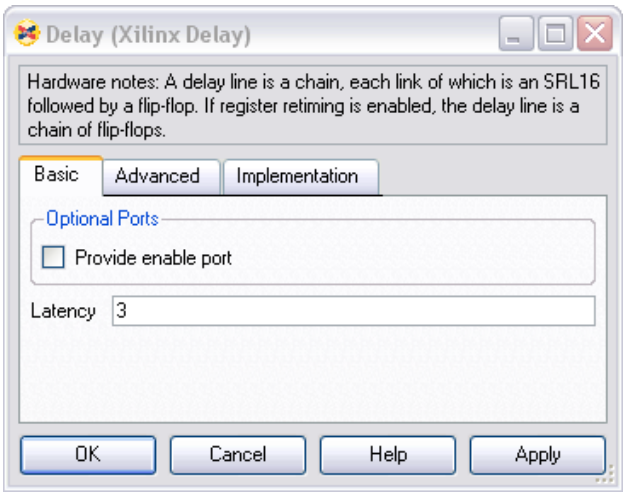

Figura 32. Configuración para el ejemplo funcionamiento del bloque Xilinx Delay.

El sistema se configura con una latencia de 3.

Señales

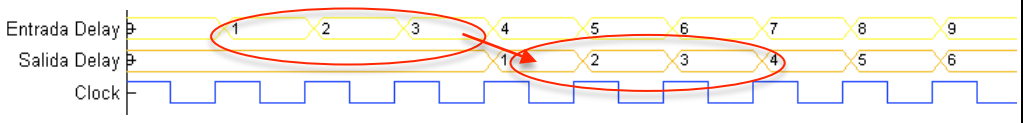

Figura 33.Señales de Ejemplo funcionamiento del bloque Xilinx Delay.

3.4.7 Bloque Xilinx Paralelo to Serial

Descripción

El bloque Xilinx Parallel to Serial (figura 34) toma una palabra a la entrada de cualquier tamaño y la divide en N palabras de salida multiplexadas en tiempo, siendo N la relación entre el número de bits de la palabra de entrada y la de salida.

Figura

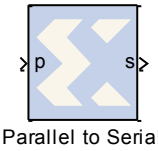

Figura 34. Bloque Xilinx Parallel to Serial

Puertos de entrada y salida

Entrada

E-1 ***Input*** - Entrada

Flujo de palabras interpretados como datos en paralelo

Salida

S-1 ***Output*** - Salida

Flujo palabras de menor tamaño, ordenados en serie.

Configuración Básica

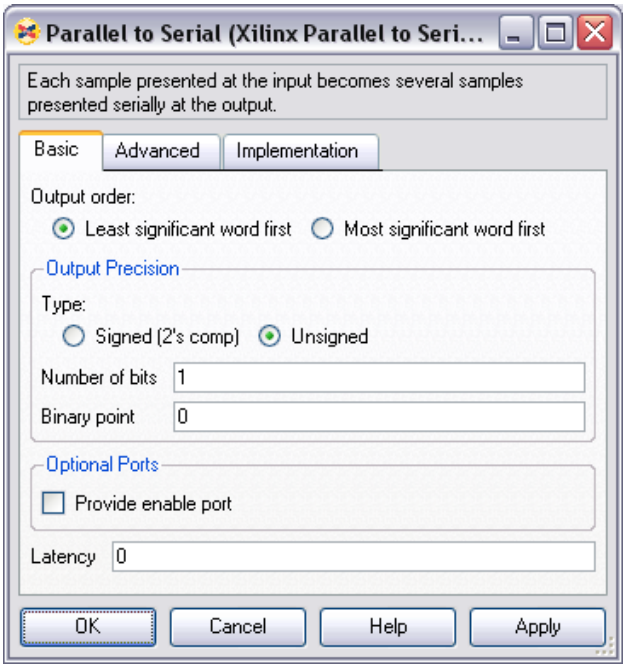

Figura 35. Ventana de Configuración básica del bloque Xilinx Parallel to Serial

Parámetros

1	***Output order*** - Orden de salida
	Orden deseado para los datos de salida. Aplica a nivel de palabra, la que posea los datos más o menos significativos de la palabra de entrada. Puede ser: **Least significant word first** o **Most significant word first**
2	***Arithmetic Type*** - Tipo de aritmética
	Utilización o no de signo (complemento a dos) para los datos de salida.
3	***Number of bits*** - Número de bits
	Tamaño de la palabra de salida. Dicho número debe ser un divisor entero del tamaño de la palabra de entrada.
4	***Binary point*** - Punto binario
	Especifica la posición del punto binario en la palabra de salida
5	***Provide Enable Port*** - Proporcionar Puerto de Habilitación
	De activarse esta opción, el bloque mostrará un puerto adicional de habilitación. La señal que controle este puerto ha de ser de tipo booleano.
6	***Latency*** - Latencia
	Este bloque no introduce latencia, pero es posible especificar un número de períodos de muestreo si se desea retardar los datos.

Observaciones

- La ejecución de una operación paralelo a serie necesariamente multiplica la tasa binaria en la misma proporción que se reduce el tamaño de la palabra (N), lo que puede afectar a la tasa de muestreo general del sistema.

Ejemplo de Funcionamiento

Modelo

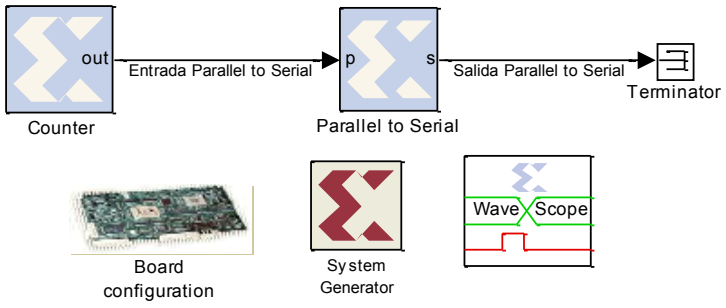

Figura 36. Modelo de ejemplo de funcionamiento del bloque Xilinx Parallel to Serial.

Descripción

Este modelo consta de:

- Una fuente de datos, papel desempeñado por bloque Xilinx Counter configurado como contador libre de 4 bits sin signo.
- Un bloque Xilinx Parallel to Serial ordenando la entrada con la palabra menos significativa primero y una palabra de salida de 2 bits.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Este modelo esta formado por una fuente de datos de 4 bits y una salida de 2 bits, siendo divididos por el bloque *Xilinx Parallel to Serial* (tal y como se observa en las figuras 37 y 38.

Configuración

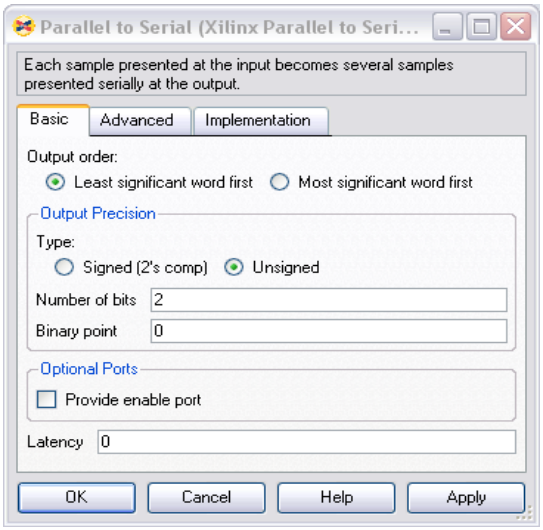

Figura 37. Configuración para el ejemplo funcionamiento del bloque Xilinx Parallel to Serial.

El bloque se configura para que entregue la palabra menos significativa primero, no utilice signo. La palabra de salida se de define a 2 bits sin punto binario. No se solicita puerto de habilitación y la latencia se deja a 0 (por defecto).

Señales

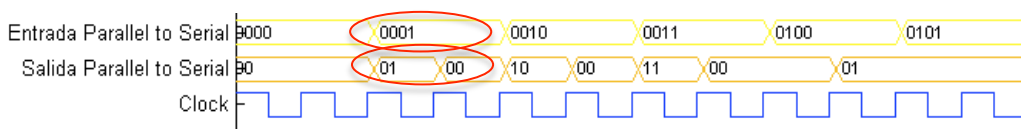

Figura 38.Señales de Ejemplo funcionamiento del bloque Xilinx Parallel to Serial.

3.4.8 Bloque Xilinx Serial to Parallel

Descripción

El bloque *Xilinx Serial to Parallel* (figura 39) toma una serie de entradas de cualquier tamaño y proporciona una única salida con una palabra de tamaño múltiplo del tamaño de la entrada.

Figura

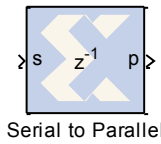

Figura 39. Bloque Xilinx Serial to Parallel

Puertos de entrada y salida

Entrada

E-1 ***Input*** - Entrada

Datos, interpretados como un flujo en serie, a partir de los cuales se quiere generar la palabra de mayor tamaño

Salida

S-1 ***Output*** - Salida

Flujo de palabras resultado de agrupar las palabras de la entrada.

Configuración Básica

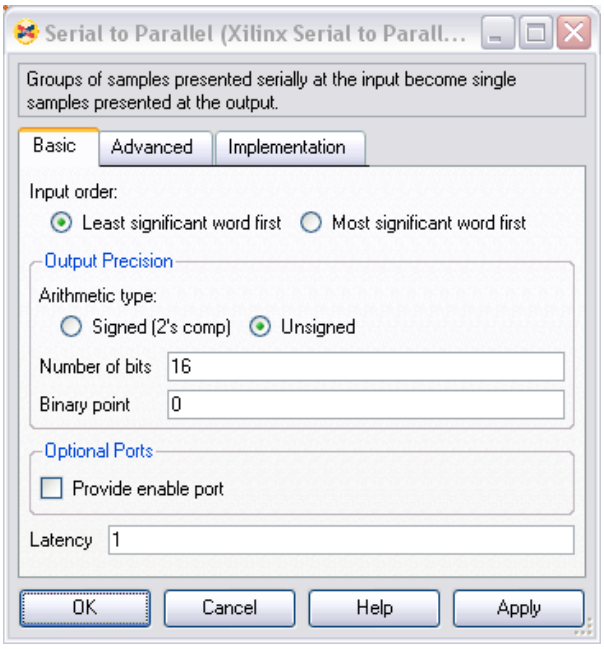

Figura 40. Ventana de Configuración básica del bloque Xilinx Serial to Parallel

Parámetros

1	***Input order*** - Orden de entrada
	Indica como se deber interpretar las palabras a la entrada a la hora de colocarlas en paralelo. Aplica a nivel de palabra. Indica si la primera palabra del flujo aporta los datos más o menos significativos de la palabra creada. Puede ser: **Least significant word first** o **Most significant word first**
2	***Arithmetic Type*** - Tipo de aritmética
	Utilización o no de signo (complemento a dos) para los datos de salida.
3	***Number of bits*** - Número de bits
	Tamaño de la palabra de salida. Dicho número debe ser un múltiplo entero del tamaño de la palabra de entrada.
4	***Binary point*** - Punto binario
	Especifica la posición del punto binario en la palabra de salida
5	***Provide Enable Port*** - Proporcionar Puerto de Habilitación
	De activarse esta opción, el bloque mostrará un puerto adicional de habilitación. La señal que controle este puerto ha de ser de tipo booleano.
6	***Latency*** - Latencia
	Este bloque introduce latencia, la necesaria para acumular todas las palabras en la entrada necesarias para formar la palabra de salida. La latencia se define desde el punto de vista de la salida, con lo que la latencia mínima es uno. Es posible especificar un número adicional de períodos de muestreo si se desea retardar los datos.

Observaciones

- La ejecución de una operación serie a paralelo necesariamente reduce la tasa binaria en la misma proporción que se reduce el tamaño de la palabra (N).

Ejemplo de Funcionamiento

Modelo

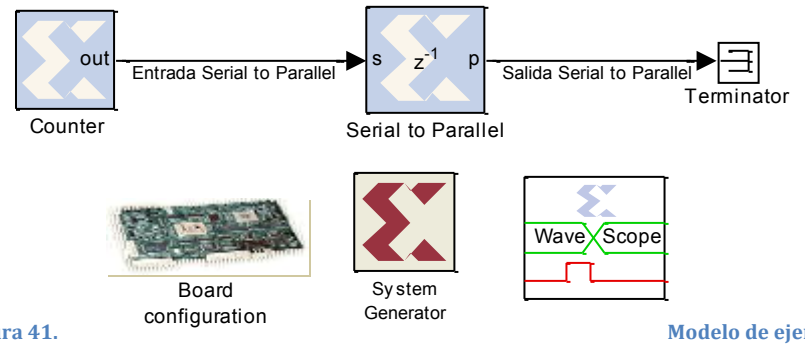

Figura 41. Modelo de ejemplo de funcionamiento del bloque Xilinx Serial to Parallel.

Descripción

Este modelo consta de:

- Una fuente de datos, papel desempeñado por bloque Xilinx Counter configurado como contador libre de 2 bits sin signo.
- Un bloque Xilinx Serial to Parallel ordenando la entrada con la palabra menos significativa primero y una palabra de salida de 4 bits.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Este modelo esta formado por una fuente de datos de 2 bits y una salida de 4 bits, siendo agrupados por el bloque Xilinx Serial to Parallel (tal y como se observa en las figuras 42 y 43.

Configuración

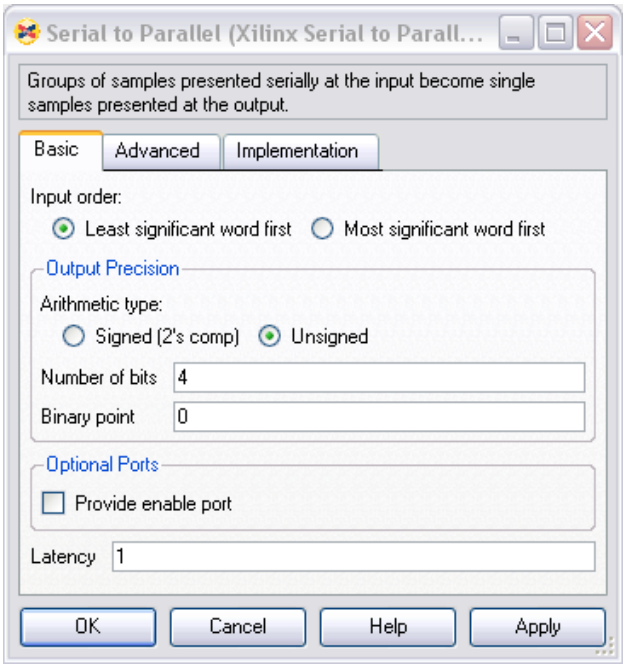

Figura 42. Configuración para el ejemplo funcionamiento del bloque Xilinx Serial to Parallel

El bloque se configura para que interprete que la palabra menos significativa es la primera, no utilice signo. La palabra de salida se de define a 4 bits sin punto binario. No se solicita puerto de habilitación y la latencia se deja a 1 (por defecto).

Señales

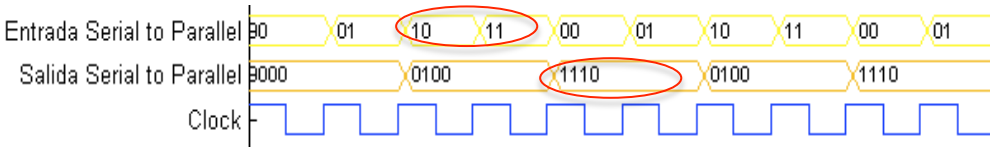

Figura 43.Señales de Ejemplo funcionamiento del bloque Xilinx Serial to Parallel.

3.4.9 Bloque Xilinx BitBasher

Descripción

El bloque Xilinx BitBasher (figura 44) permite realizar operaciones a nivel de bit como extraer, concatenar y repetir.

Figura

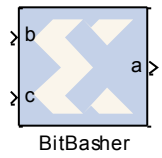

Figura 44. Bloque Xilinx BitBasher

Puertos de entrada y salida

Entrada

E-1 ***b,c,…***

Datos sobre los cuales se desea operar. El nombre del puerto es el empleado en la definición de la operación.

Salida

S-1 ***a,…***

Datos resultado de aplicar las operaciones definidas. Se pueden definir hasta cuatro operaciones, que producen hasta cuatro puertos de salida. El nombre empleado es el empleado en la definición de la operación.

Configuración Básica

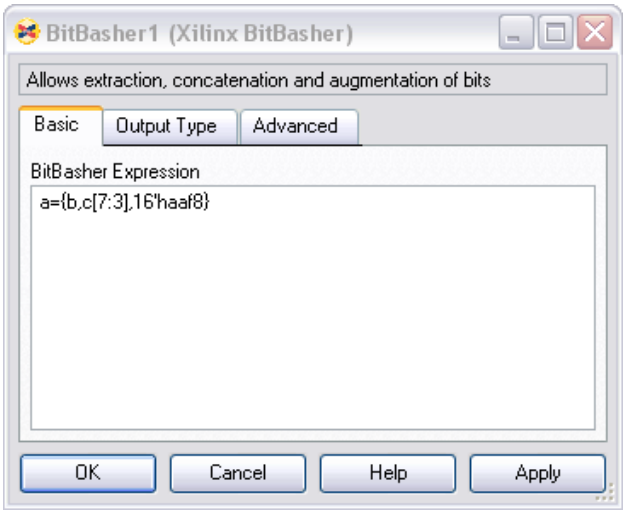

Figura 45. Ventana de Configuración básica del bloque Xilinx BitBasher. Primera Pestaña

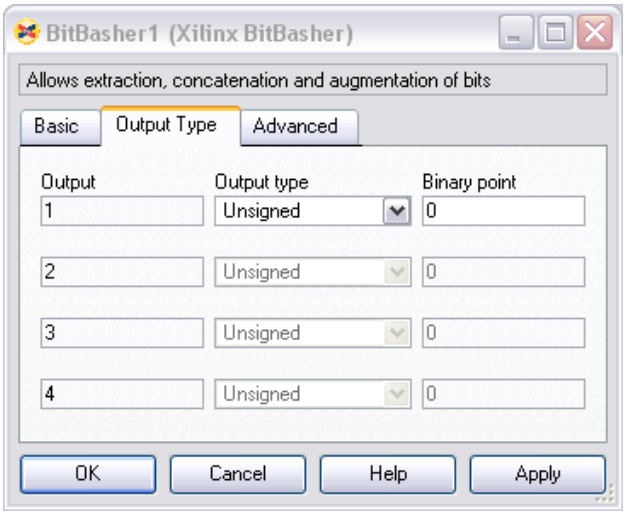

Figura 46. Ventana de Configuración del bloque Xilinx Bitbasher. Segunda Pestaña

Parámetros	
1	***Basic*** - Básico ***BitBasher Expression*** - Expresión del BitBasher
	Expresión (o Expresiones) que definen la operación (u operaciones) a realizar. Se pueden definir varias expresiones (hasta 4), en líneas distintas. Este campo se encuentra en la primera pestaña, *Basic*.
2	***Output Type*** - Tipo de Salida ***Output*** - Salida
	Indica el número de puerto sobre el que se van a configurar los siguientes parámetros (4 y 5). Este campo se encuentra en la segunda pestaña, *Output Type*.
3	***Output Type*** - Tipo de Salida ***Output Type*** - Tipo de Salida
	Indica el tipo de aritmética que se desea usar para el puerto de salida señalado en el parámetro 3, con signo (en complemento a dos) o sin signo. Este campo se encuentra en la segunda pestaña, *Output Type*.
4	***Output Type*** - Tipo de Salida **Binary Point** - Punto Binario
	Especifica la posición del punto binario en la palabra de salida, para el puerto señalado en el parámetro 3 . Este campo se encuentra en la segunda pestaña, *Output Type*.

Observaciones

- El bloque permite crear hasta cuatro expresiones distintas, que se traducen en hasta 4 puertos de salida . El sistema crea tantos puertos como expresiones se hayan definido.
- Las operaciones se describen mediante sintaxis Verilog.

Ejemplo de Funcionamiento

Modelo

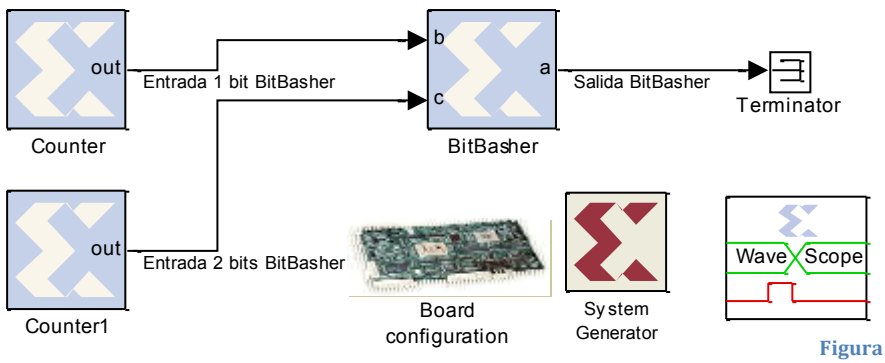

Figura 47.

Modelo de ejemplo de funcionamiento del bloque Xilinx BitBasher.

Descripción

Este modelo consta de:

- Dos fuentes de datos, papel desempeñado por sendos bloques Xilinx Counter configurados como contadores libres de 1 y 2 bits sin signo respectivamente.
- Un bloque Xilinx Bitbasher definido para concatenar y reordenar las entradas, situando el punto binario en el bit 1.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Este modelo esta formado por dos fuentes de datos de 1 y 2 bits y una salida de 3 bits con el punto binario en el bit 1, siendo ejecutada esta operación por el bloque *Xilinx Bitbasher,* tal y como se observa en las figuras 48, 49 y 50.

Configuración

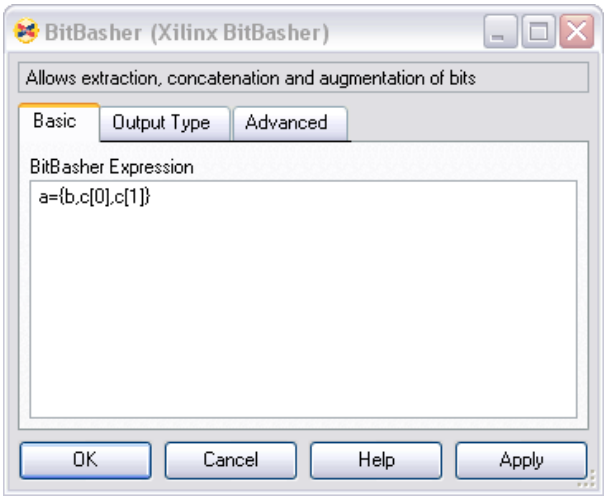

Figura 48. Configuración de la ventana básica para el ejemplo funcionamiento del bloque Xilinx BitBasher

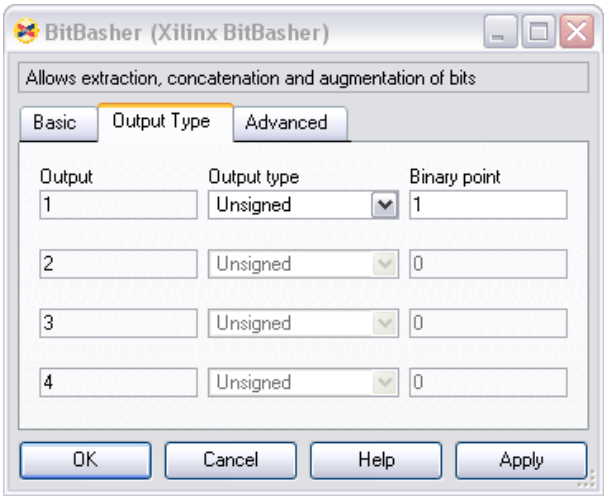

Figura 49. Configuración de la ventana de tipo de salida, para el ejemplo funcionamiento del bloque Xilinx BitBasher

El bloque se define mediante la expresión a= {b,c[0],c[1]}, que indica como se deben concatenar las palabras de entrada.Primer el bit de la primera fuente (palabra de un solo bit), a continuación el bit menos significativo de la segunda fuente (palabra de 2 bits) y por último el bit más significativo de la segunda fuente (palabra de 2 bits). En la segunda pestaña se configura la única salida sin signo y con el punto binario en el bit 1 (que se corresponde con el bit más signficativo de la segunda fuente).

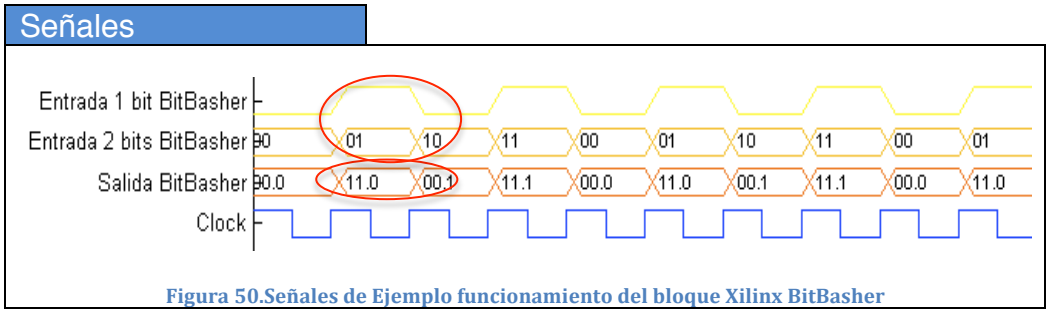

Figura 50.Señales de Ejemplo funcionamiento del bloque Xilinx BitBasher

3.4.10 Bloque Xilinx Mcode

Descripción

El Bloque Xilinx Mcode (figura 51) permite generar código VHDL/Verilog a partir de código MATLAB® generado por el desarrollador.

Figura

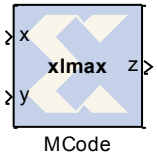

Figura 51. Bloque Xilinx Mcode

Puertos de entrada y salida

Entrada

E-1 ***x,y,…***

Cada uno de los parámetros de entrada de las función que se implementa. Se genera un puerto por cada parámetro que se especifique en la declaración de la función, que sigue el estándar MATLAB®: function [parámetros de salida] = nombre (parámetros de entrada)

Salida

S-1 ***z,…***

Cada uno de los parámetros de salida de la función que se implementa. Se genera un puerto por cada parámetro que se especifique en la declaración de la función, que sigue el estándar MATLAB®: function [parámetros de salida] = nombre (parámetros de entrada)

Configuración Básica

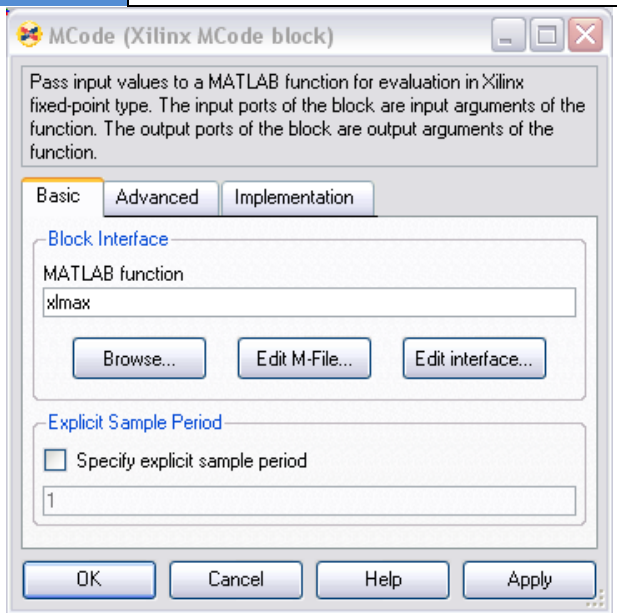

Figura 52. Ventana de Configuración básica del bloque Xilinx MCode. Primera Pestaña

Parámetros

1	***MATLAB function*** - función MATLAB
	Nombre del código de extensión que debe ejecutar el bloque, debe tener la extensión .m
2	***Browse*** - Navegar
	Botón para abrir la ventana de exploración para localizar el archivo que contiene el código.
3	***Edit M-file*** – Editar fichero tipo M
	Abre el editor de MATLAB® y permite la edición del código
4	***Edit Interface*** – editar interfaz
	Muestra la lista de entradas y salidas de la función con sus valores constantes, si los tuvieran.

Observaciones

- Este bloque permite realizar operaciones aritméticas y lógicas relativamente complejas, de forma más sencilla que empleando un conjunto de bloques Xilinx equivalente. También resulta útil para el diseño de máquinas de estado.
- El bloque Xilinx Mcode admite un subconjunto de operaciones MATLAB®. Principalmente las relacionadas con las funciones anteriormente descritas, operaciones aritméticas, lógicas, estados, etc.
- Todas las entradas y salidas han de ser de tipo punto fijo.
- El bloque tiene que tener al menos una salida.
- El bloque Xilinx Mcode posee herramientas de depuración de código

Ejemplo de Funcionamiento

Modelo

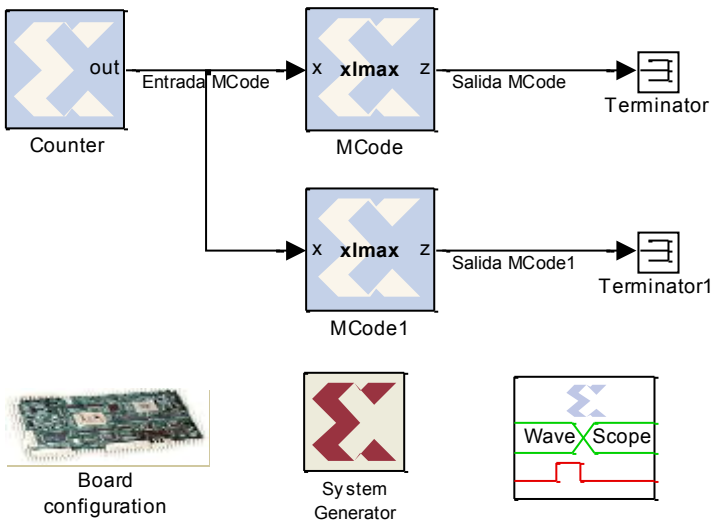

Figura 53. Modelo de ejemplo de funcionamiento del bloque Xilinx Mcode.

Descripción

Este modelo consta de:

- Una fuente de datos que consiste en un bloque Xilinx Counter configurado como contador libre de 3 bits sin signo. Esta fuente es la entrada a ambos bloques Xilinx MCode.
- Dos bloques Xilinx MCode que ejecutan la misma función (comparación de la entrada con un valor fijo mostrando a la salida el máximo) con diferentes parámetros (umbral de valor 2 y 5).
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida de los bloques.

Los bloques M-Code se han configurado para funcionar como comparadores con umbrales diferentes. El código para ambos es el mismo y se diferencian en los parámetros. Cada bloque cuenta con dos parámetros de entrada definidos y uno de salida. Una de las entradas se ha fijado (umbral fijo), por lo que no aparece como puerto externo. La salida es el máximo entre la entrada variable (puerto externo) y el umbral fijo. La configuración puede observarse en las figuras 54, 55, 56, 57 y 58.

Configuración

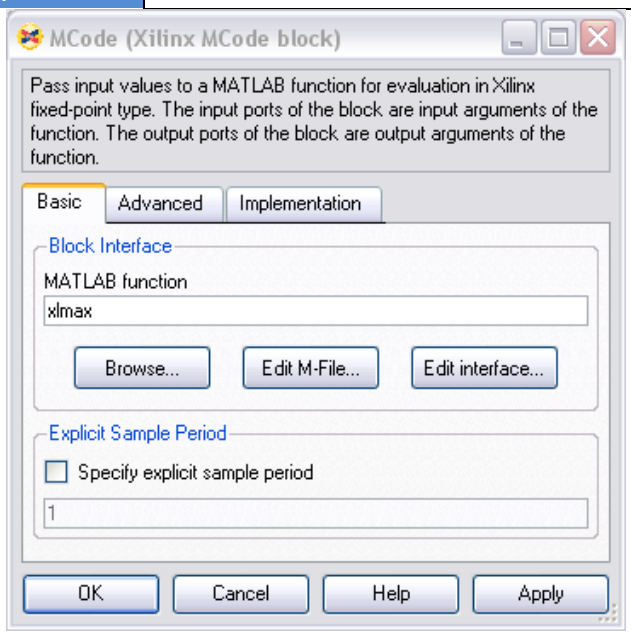

Figura 54. Configuración del bloque para el ejemplo funcionamiento del bloque Xilinx Mcode

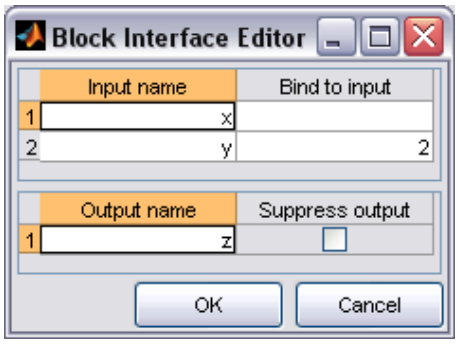

Figura 55. Configuración de la Interfaz del primer bloque Mcode para el ejemplo funcionamiento del bloque Xilinx Mcode

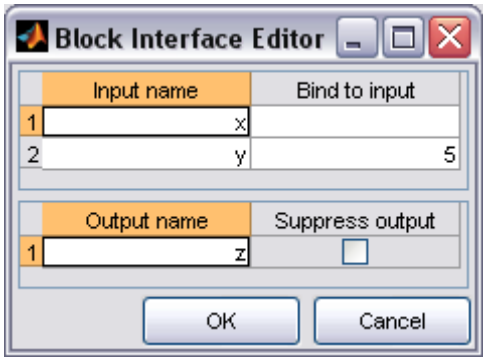

Figura 56. Configuración de la Interfaz del segundo bloque Mcode para el ejemplo funcionamiento del bloque Xilinx Mcode

```
function z = xlmax(x, y)
```

```
if x > y
z = x;
else
z = y;
end
```

Figura 57. Código implementado en ambos bloques Mcode para el ejemplo funcionamiento del bloque Xilinx Mcode

Señales

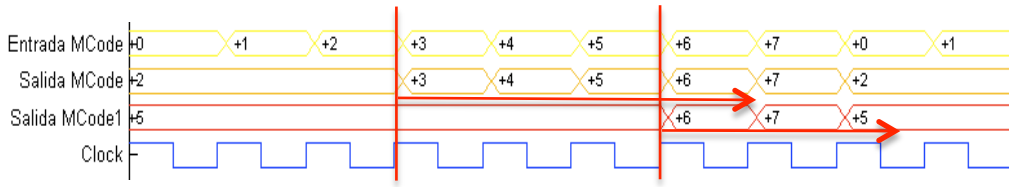

Figura 58.Señales de Ejemplo funcionamiento del bloque Xilinx Mcode

3.4.11 Bloque Xilinx Logical

Descripción

El Bloque Xilinx Logical (figura 59) se emplea en la realización de operaciones lógicas a nivel de bit con datos que contengan punto binario.

Figura

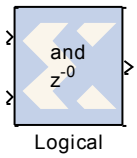

Figura 59. Bloque Xilinx Logical

Puertos de entrada y salida

Entrada

E-1 ***Input*** - Entrada

Datos sobre los que se va a realizar la operación. El bloque admite hasta un máximo de 1024 entradas.

Salida

S-1 ***Output*** - Salida

Resultado de la operación lógica configurada sobre los datos de entrada.

Configuración Básica

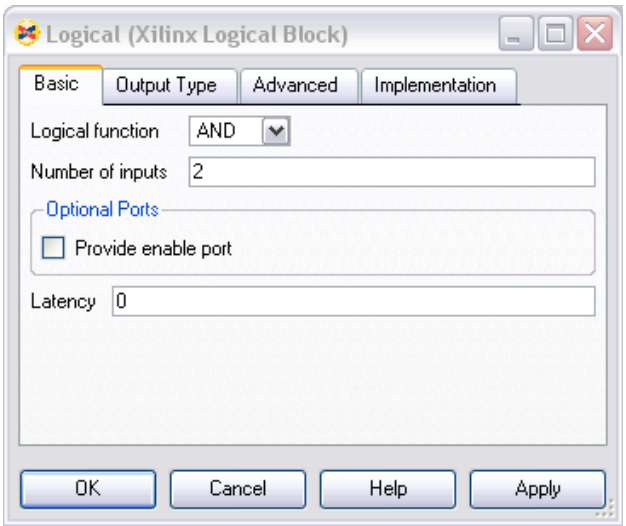

Figura 60. Ventana de Configuración básica del bloque Xilinx Logical (primera pestaña)

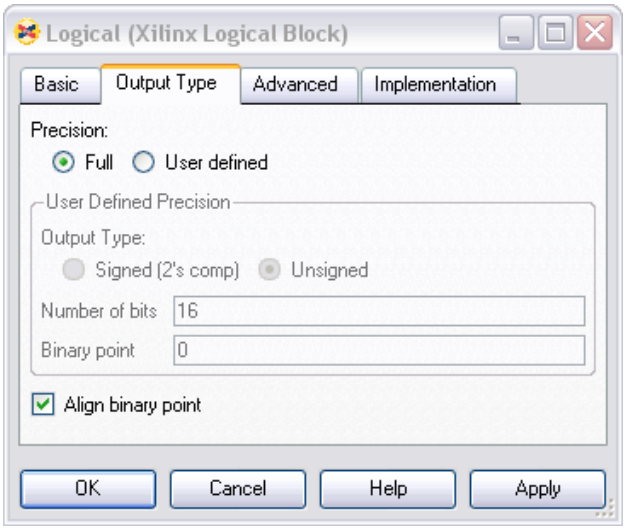

Figura 61. Ventana de Configuración del Tipo de Salida del bloque Xilinx Logical (segunda pestaña)

Parámetros	
1	***Basic*** - Básico ***Logical function*** – Función Lógica
	Operación lógica que se desea implementar. Las opciones son: *AND, NAND, OR, NOR, XOR* y *XNOR*.
2	***Basic*** - Básico ***Number of inputs*** – Número de entradas
	Número de entradas con las que contará el bloque, desde 1 hasta 1024.
3	***Output Type*** - Tipo de Salida ***Precision*** - Precisión
	Precisión, definida en número de bits y posición del punto binario y criterio de signo (sin signo o con signo complemento a 2) En caso de que se seleccione *full* la precisión será la necesaria para representar sin error el valor resultante de la operación lógica realizada.
4	***Output Type*** - Tipo de Salida ***Align binary point*** - Alinear punto binario
	En caso de seleccionarse, los puntos binarios de las entradas son alineados de forma automática. En caso contrario, las entradas deben tener el punto binario en la misma posición.

Observaciones

- Los puntos binarios de las entradas deben estar alineados para poder realizar la operación. En su caso, las entradas se completan con ceros y se extienden en signo a tal efecto.

Ejemplo de Funcionamiento

Modelo

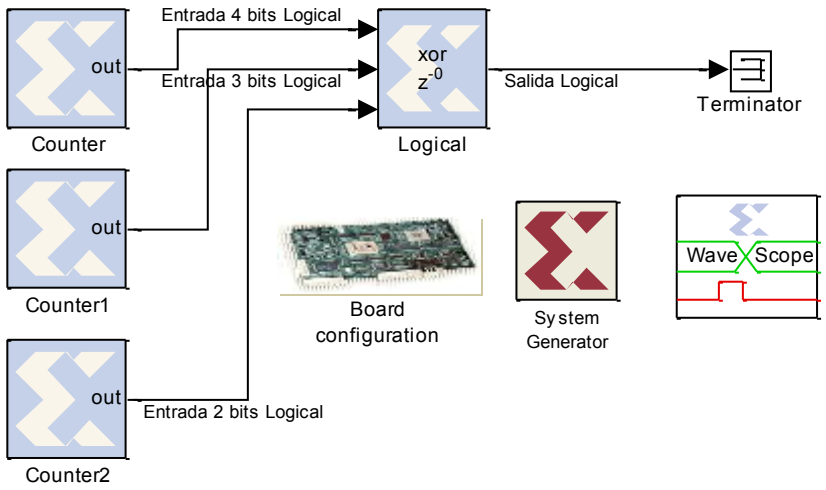

Figura 62. Modelo de ejemplo de funcionamiento del bloque Xilinx Logical.

Descripción

Este modelo consta de:

- Tres fuentes de datos consistentes, para cada una de las cuales se emplea un bloque Xilinx Counter configurado como contadores libre de 4, 3 y 2 bits, respectivamente. El número de inicio y la posición del punto binario también son distintos para cada uno de ellos. Estas fuentes son la entrada al bloque Xilinx Logical.
- Un bloque Xilinx Logical configurado para realizar la función XOR (que es asociativa) , empleando la máxima precisión a la salida.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida de los bloques.

Se ha configurado el modelo para realizar la función XOR (OR exclusivo) de las tres fuentes datos. La operación se realiza a nivel de bit, por lo que el resultado es comparable al de realizar un control de paridad de las tres fuentes (la salida será un uno cuando la entrada tenga un número impar de unos)

El bloque Xilinx Logical está configurado ara alinear automáticamente el punto binario y para proporcionar la máxima precisión de salida (`full`)

La configuración puede observarse en las figuras 63, 64, y 65.

Configuración

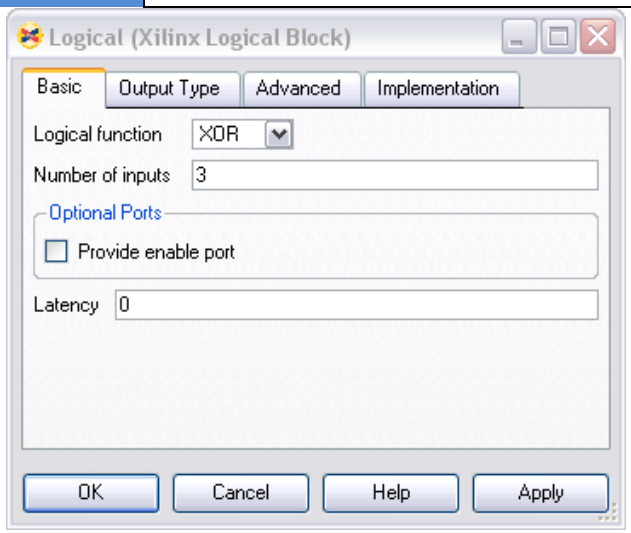

Figura 63. Configuración del bloque para el ejemplo funcionamiento del bloque Xilinx Logical

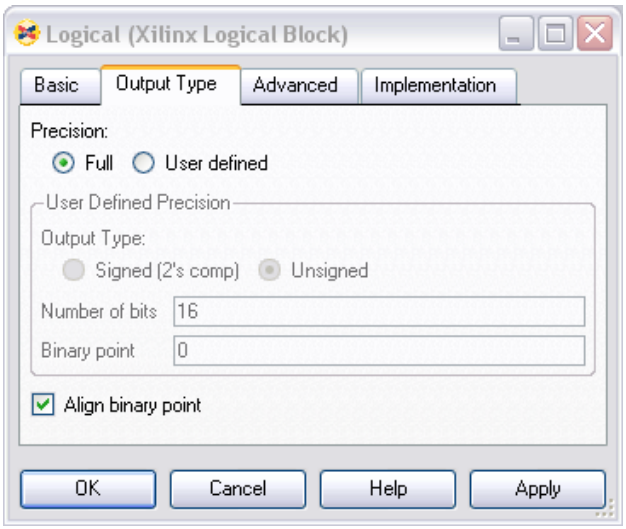

Figura 64. Configuración del tipo de salida para el ejemplo funcionamiento del bloque Xilinx Logical

Señales

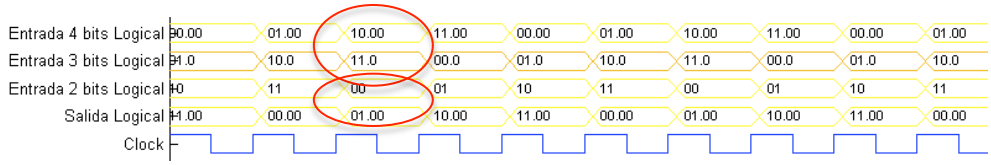

Figura 65.Señales de Ejemplo funcionamiento del bloque Xilinx Logical

3.4.12 Bloque Xilinx Shift

Descripción

El Bloque Xilinx Shift (figura 66) se emplea para realizar desplazamientos lógicos a derecha o izquierda, sin modificar la posición del punto binario.

Figura

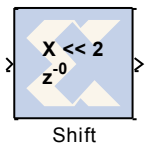

Figura 66. Bloque Xilinx Shift

Puertos de entrada y salida

Entrada

E-1 ***Input*** - Entrada

Datos sobre los que se desea efectuar el desplazamiento lógico.

Salida

S-1 ***Output*** - Salida

Resultado de la operación de desplazamiento lógico. Palabras con el punto binario en la misma posición que los datos de entrada y los datos desplazados tantos bits como se haya indicado.

Configuración Básica

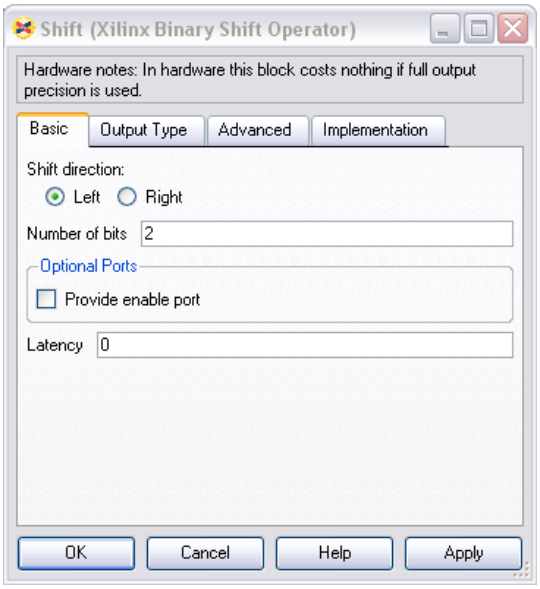

Figura 67. Ventana de Configuración básica del bloque Xilinx Logical (primera pestaña)

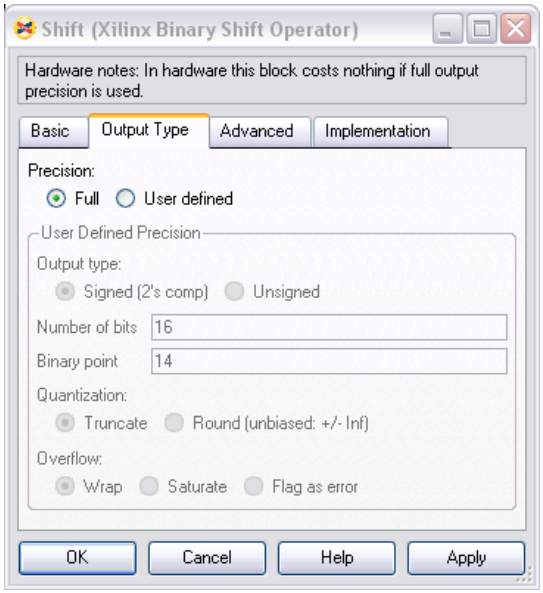

Figura 68. Ventana de Configuración del tipo de salida del bloque Xilinx Logical (segunda pestaña)

Parámetros

1	***Basic*** - Básico ***Shift Direction*** – Dirección de desplazamiento
	Sentido en el que se desea realizar el desplazamiento, izquierda o derecha.
2	***Basic*** - Básico ***Number of bits*** – Número de bits
	Número de bits que se van a desplazar-
3	***Output Type*** - Tipo de Salida ***Precision*** - Precisión
	Precisión, definida en número de bits y posición del punto binario y criterio de signo (sin signo o con signo complemento a 2) En caso de que se seleccione `full` la precisión será la necesaria para representar sin error el valor resultante de la operación de desplazamiento lógico realizada.

Observaciones

- Independientemente de la dirección del desplazamiento, las nuevas posiciones son rellenadas con ceros.
- El bloque posee puerto de entrada de habilitación (`enable port`) opcional. Su entrada deber ser de tipo booleana.

Ejemplo de Funcionamiento

Modelo

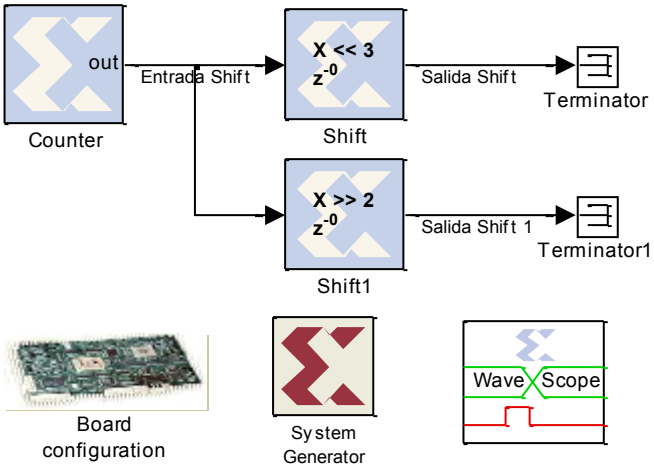

Figura 69. Modelo de ejemplo de funcionamiento del bloque Xilinx Shift.

Descripción

Este modelo consta de:

- Una fuente de datos para la que se emplea un bloque Xilinx Counter configurado como contador libre de 8 bits con punto binario en el segundo bit. El contador inicia la cuenta en 16. Esta fuente alimenta a los dos bloques Xilinx Shift.
- Dos bloques Xilinx Shift. Uno de ellos realizará un desplazamiento lógico de tres bits a la izquierda y el otro de dos bits a la derecha. En lugar de configurar este último con un desplazamiento a la derecha convencional, se hará con un desplazamiento lógico a la izquierda con un número negativo (-2) de bits, por lo que se conseguirá el mismo efecto.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida de los bloques.

El sistema de ejemplo desplaza las palabras provenientes de la fuente de datos de dos formas distintas.

El primer bloque desplaza los datos hacia la izquierda, hacía el bit más significativo (operación equivalente a multiplicar por 2) rellenando con ceros por la derecha (bit menos significativo). Los bits que se salen por la izquierda de la palabra son descartados.

El segundo bloque desplaza los bits a la derecha, hacia el bit menos significativo (operación equivalente a dividir por 2) rellenando con ceros por la izquierda (bit más

significativo). Los bits que se salen por la derecha de la palabra son descartados.

Se puede observar que el punto binario permanece en el mismo punto.

La configuración puede observarse en las figuras 70, 71, 72 y 73.

Configuración

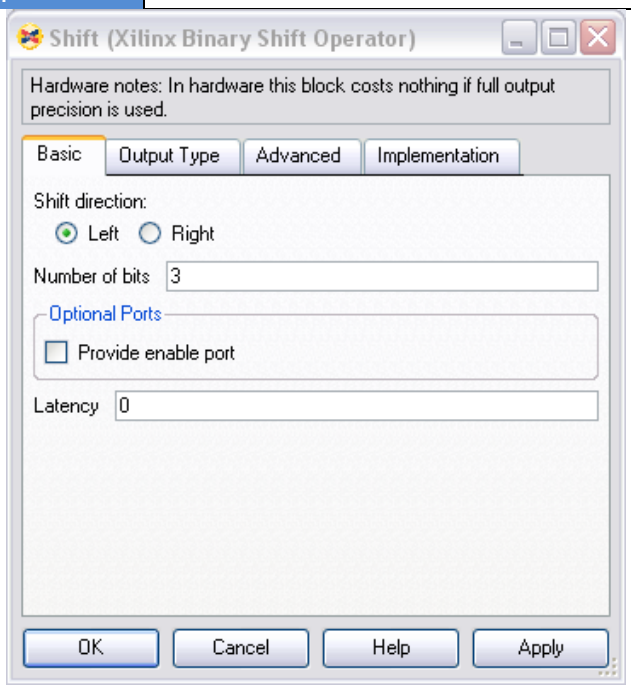

Figura 70. Configuración del bloque 1 para el ejemplo funcionamiento del bloque Xilinx Shift

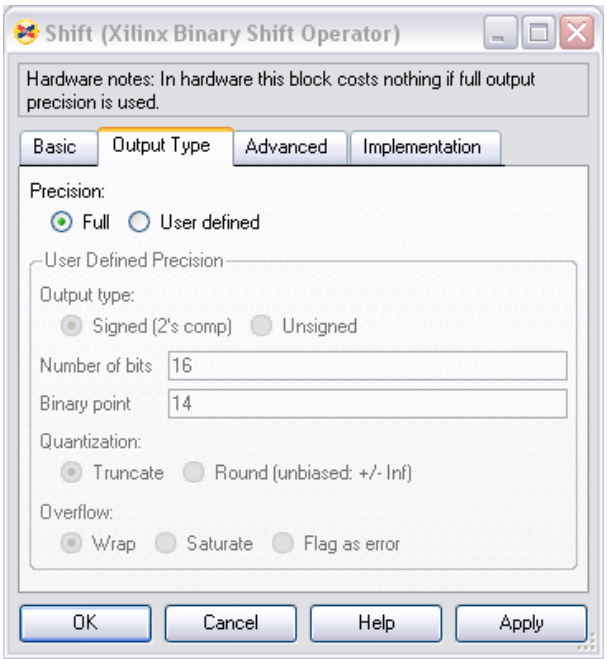

Figura 71. Configuración del tipo de salida del bloque 1 para el ejemplo funcionamiento del bloque Xilinx shift

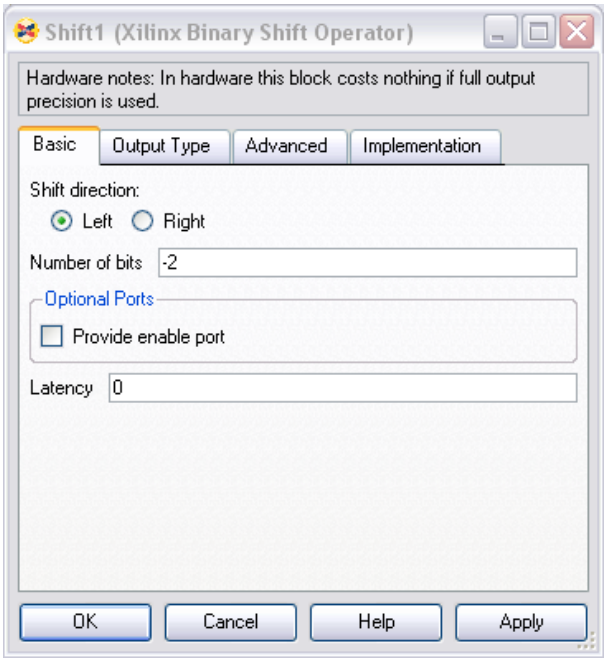

Figura 72. Configuración del bloque 2 para el ejemplo funcionamiento del bloque Xilinx Shift

Señales

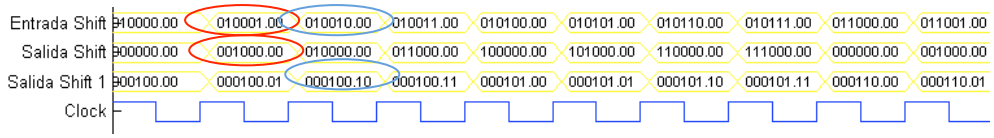

Figura 73.Señales de Ejemplo funcionamiento del bloque Xilinx Shift

3.4.13 Bloque Xilinx Constant

Descripción

El Bloque Xilinx Shift (figura 74) genera una constante. Esta puede consistir en un valor booleano, un valor con punto fijo o una instrucción DSP48.

Figura

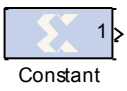

Figura 74. Bloque Xilinx Constant

Puertos de entrada y salida

Entrada

No posee

Salida

S-1 ***Output*** - Salida

Dato de valor constante especificado.

Configuración Básica

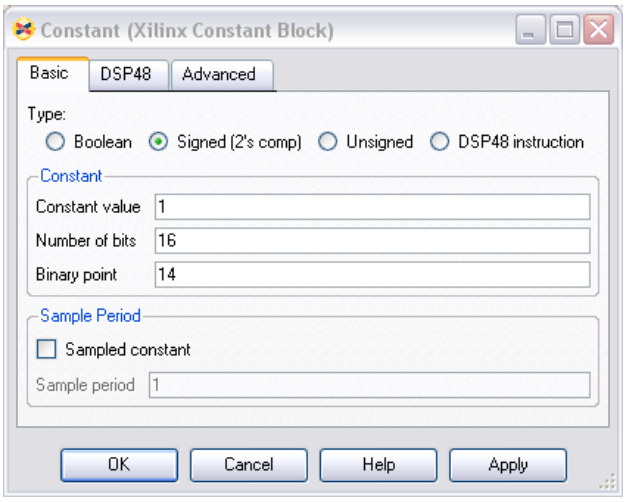

Figura 75. Ventana de Configuración básica del bloque Xilinx Constant

Parámetros

1	***Type*** – Tipo
	Tipo de la constante. Como se ha explicado, esta puede ser booleana, punto fijo con signo, punto fijo sin signo o una instrucción DSP48.
2	***Constant Value*** – Valor de la constante
	Valor de la constante. Este valor aparece en el icono del bloque.

3	***Number of bits*** – Número de bits
	Número de bits de la constante, si es de tipo punto fijo.
4	***Binary Point*** – Punto Binario
	Posición del punto binario de la constante de salida, en caso de que sea de tipo punto fijo.
5	***Sampled Constant*** – Constante Muestreada
	Asocia un periodo de muestreo a la constante, que será heredado por los bloques a los que alimente. Esto resulta importante en bloques donde las entradas deben tener el mismo periodo de muestreo.

Observaciones

- Este bloque es similar al bloque *Simulink Constant*. Sin embargo, el bloque *Xilinx Constant* puede ser empleado como entrada a otros bloques Xilinx.
- La pestaña DSP48 solo se muestra si se selecciona el tipo `DSP48 Instrucción` (instrucción DSP48) en la pestaña básica.

Ejemplo de Funcionamiento

Modelo

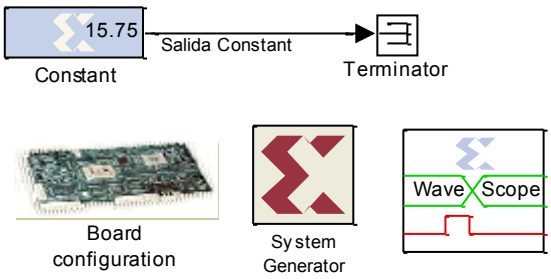

Figura 76. Modelo de ejemplo de funcionamiento del bloque Xilinx Constant

Descripción

Este modelo consta de:

- Un bloque Xilinx Constant configurado para genera una constante de tipo punto fijo con signo de valor 15.75.
- Un bloque Xilinx WaveScope para observar la forma de la señal a salida del bloque.
- El bloque Xilinx Constant se ha conectado directamente a un terminal. El

bloque del ejemplo genera la constante 15,75 (1111.11 en binario). Se ha utilizado una palabra de 12 bits con el punto binario en el cuarto bit, por lo que su representación en binario es 00001111.1100.

La configuración puede observarse en las figuras 77 y 78.

Configuración

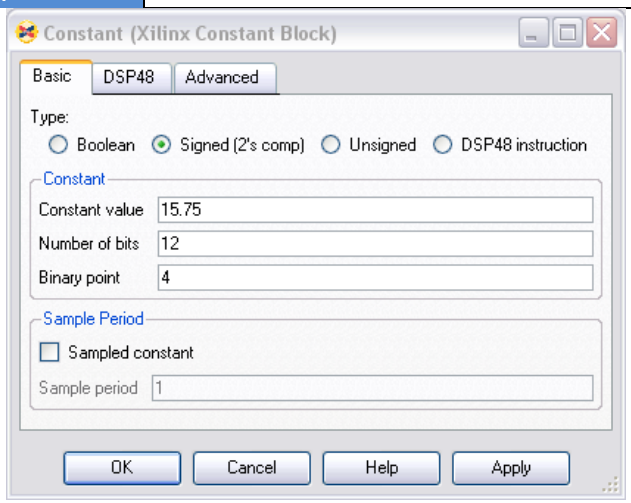

Figura 77. Configuración para el ejemplo funcionamiento del bloque Xilinx Constant

Señales

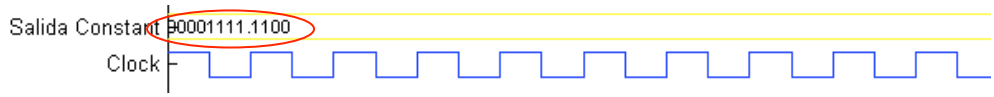

Figura 78.Señales de Ejemplo funcionamiento del bloque Xilinx Constant

3.4.14 Bloque Xilinx CMult

Descripción

El Bloque Xilinx Cmult (figura 79) multiplica la palabra a la entrada por un valor constate, que puede ser definido directamente o ser el resultado de una expresión de MATLAB®

Figura

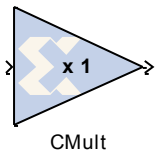

Figura 79. Bloque Xilinx CMult

Puertos de entrada y salida

Entrada

E-1	***Input*** - Entrada Datos que se desean multiplicar. El bloque admite un único puerto de entrada.

Salida

S-1	***Output*** - Salida Resultado de la multiplicación. El bloque solo admite un puerto de salida.

Configuración Básica

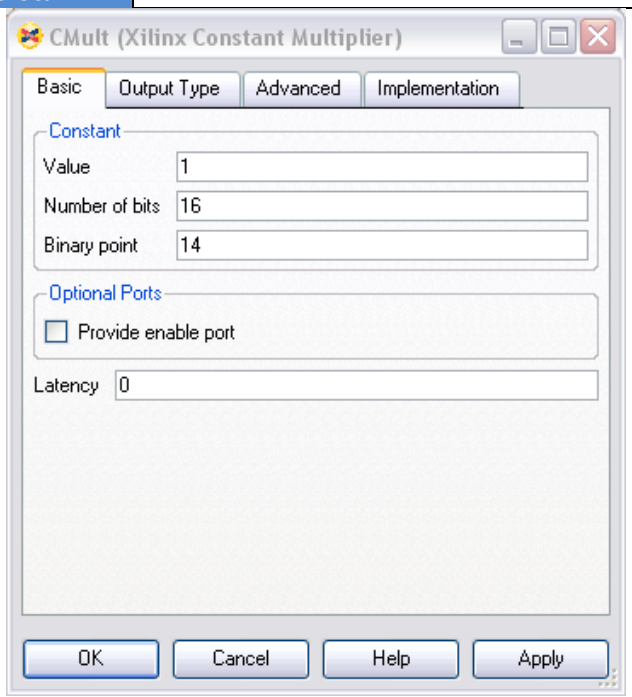

Figura 80. Ventana de Configuración básica del bloque Xilinx CMult (primera pestaña)

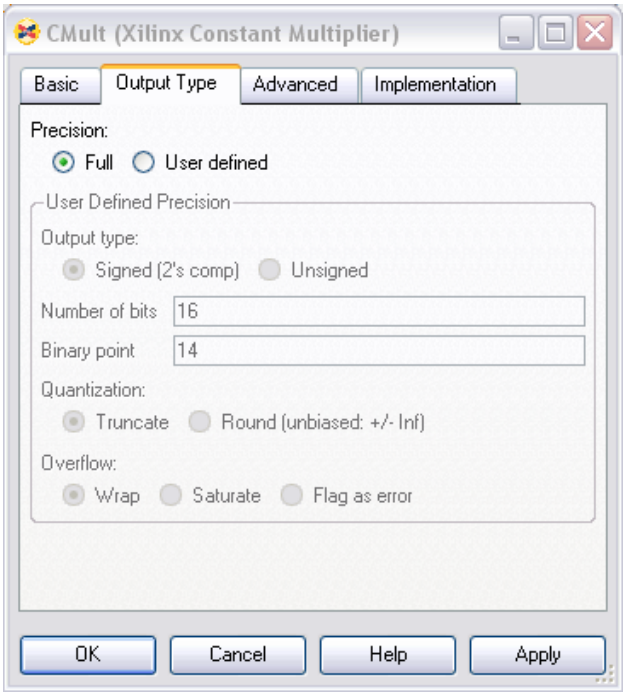

Figura 81. Ventana de Configuración del tipo de salida del bloque Xilinx CMult (segunda pestaña)

Parámetros	
1	***Basic*** - Básico ***Value***– Value
	Valor de la constante de multiplicación (constante o expresión MATLAB®). Los valores positivos se implementan sin signo y los negativos, con signo.
2	***Basic*** - Básico ***Number of bits*** – Número de bits
	Número de bits que de la constante de multiplicación.
3	***Basic*** - Básico ***Value***– Value
	Posición del punto binario de la constante de multiplicación.
4	***Output Type*** - Tipo de Salida ***Precision*** - Precisión
	Precisión, definida en número de bits y posición del punto binario y criterio de signo (sin signo o con signo complemento a 2) En caso de que se seleccione `full` la precisión será la necesaria para representar sin error el valor resultante de la operación de desplazamiento lógico realizada.

Observaciones

- En caso de que el valor de la constante no pueda ser definida con el número de bits y la posición del punto binario seleccionado, el valor es redondeado y saturado tanto como sea necesario.

Ejemplo de Funcionamiento

Modelo

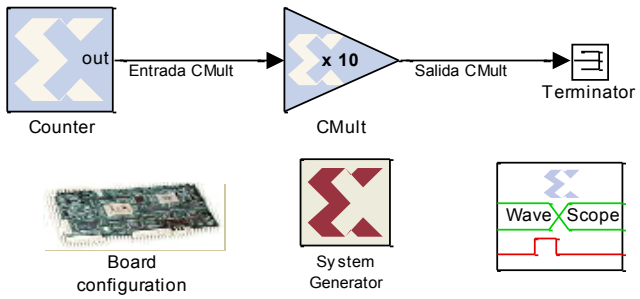

Figura 82. Modelo de ejemplo de funcionamiento del bloque Xilinx CMult

Descripción

Este modelo consta de:

- Una fuente de datos para la que se emplea un bloque Xilinx Counter configurado como contador libre de 4 bits empezando la cuenta en 11. Este bloque atacará a al entrada del bloque Xilinx CMult.
- Un bloque Xilinx CMult configurado para multiplicar por 10 la entrada. Se emplean 7 bits con el punto binario situado en el segundo bit.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

Se ha definido el bloque *Xilinx CMult* mediante un expresión MATLAB®: 5x2. Se opera con dos bits decimales. Se emplea a la salida la resolución necesaria para que no tenga que ser redondeada.

En el bloque *Xilinx WaveScope* se configura para mostrar la salida del bloque *Xilinx CMult* tanto en decimal como en binario.

La configuración puede observarse en las figuras 83 , 84 y 85.

Configuración

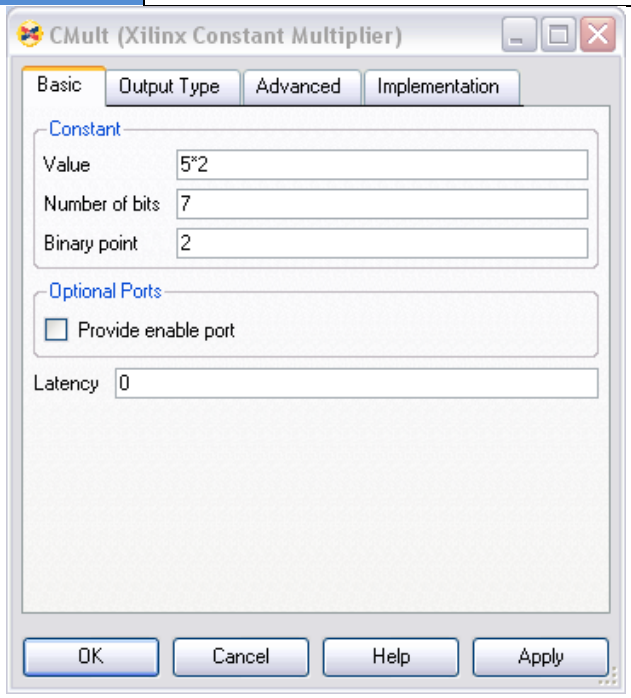

Figura 83. Configuración básica para el ejemplo funcionamiento del bloque Xilinx CMult

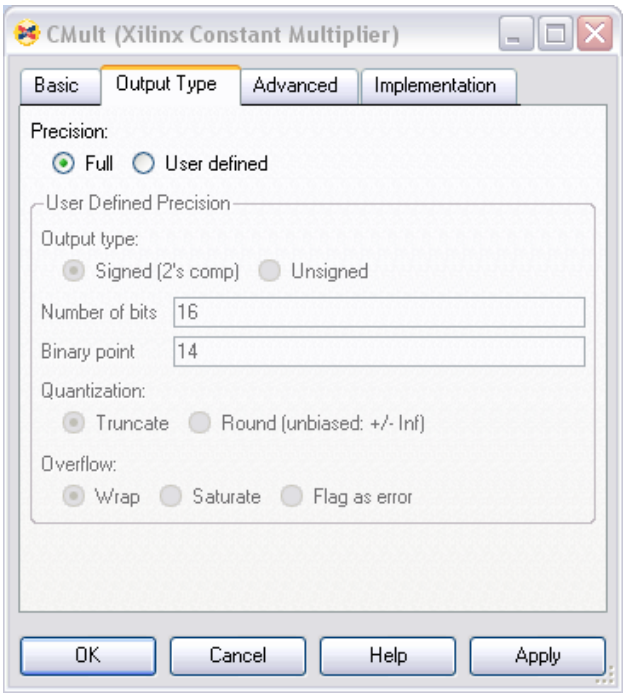

Figura 84. Configuración del tipo de salida para el ejemplo funcionamiento del bloque Xilinx CMult

Señales

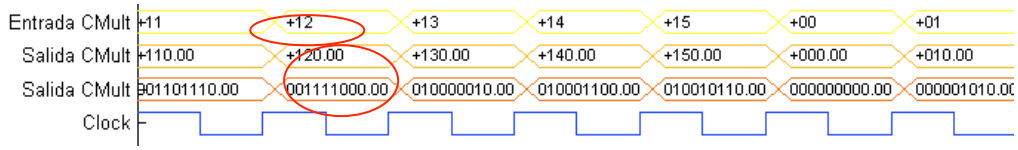

Figura 85.Señales de Ejemplo funcionamiento del bloque Xilinx Shift

3.4.15 Bloque Xilinx Mux

Descripción

El Bloque Xilinx Cmult (figura 86) implementa un multiplexor.

Figura

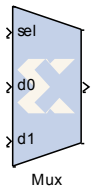

Figura 86. Bloque Xilinx CMult

Puertos de entrada y salida

Entrada

E-1 ***Input*** - Entrada

Datos que se desean multiplicar. El bloque admite un único puerto de entrada.

E-2 ***Sel*** – Selección

Selecciona la línea de entrada que se mostrará a la salida. Se define con el número de bits optimo para direccionar todas las líneas de entrada.

Salida

S-1 ***Output*** - Salida

Línea de entrada seleccionada puesta a la salida.

Configuración Básica

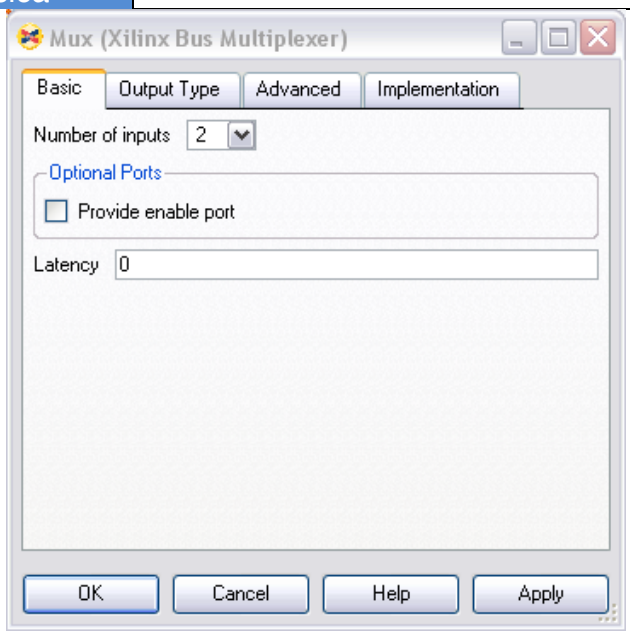

Figura 87. Ventana de Configuración básica del bloque Xilinx Mux (primera pestaña)

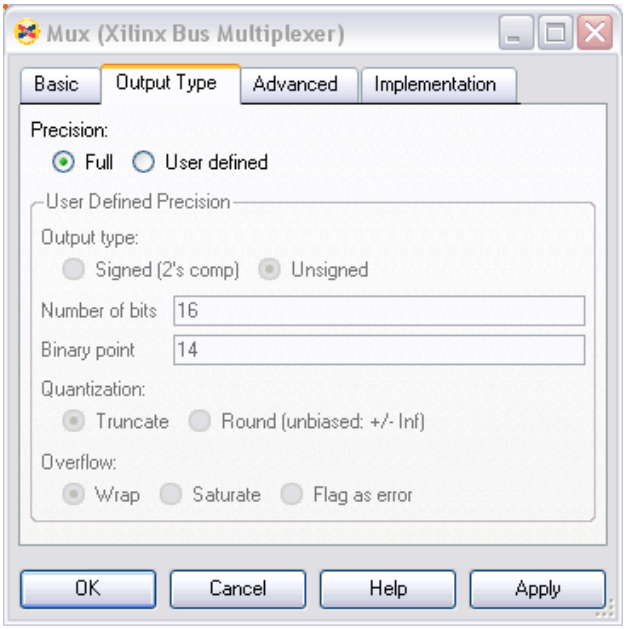

Figura 88. Ventana de Configuración del tipo de salida del bloque Xilinx Mux (segunda pestaña)

Parámetros

1	***Basic*** - Básico ***Number of inputs*** – Número de entradas
	Número de líneas de entrada al bloque, entre 2 y 32.
2	***Basic*** - Básico ***Latency*** – Latencia
	Retardo opcional entre entrada y salida. La configuración por defecto del bloque no introduce retardo.
3	***Output Type*** - Tipo de Salida ***Precision*** - Precisión
	Precisión, definida en número de bits y posición del punto binario y criterio de signo (sin signo o con signo complemento a 2) En caso de que se seleccione `full` la precisión será la necesaria para representar sin error el valor resultante de la operación de desplazamiento lógico realizada.

Observaciones

- El bloque posee puerto de entrada de habilitación (`enable port`) opcional. Su entrada deber ser de tipo booleana.

Ejemplo de Funcionamiento

Modelo

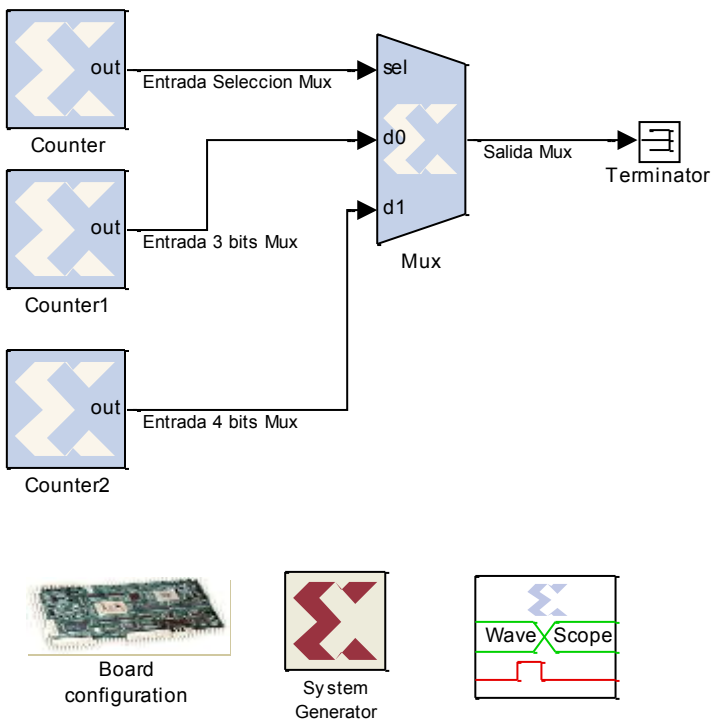

Figura 89. Modelo de ejemplo de funcionamiento del bloque Xilinx Mux

Descripción

Este modelo consta de:

- Dos fuente de datos para la que se emplean dos bloques Xilinx Counter configurados como contadores libres de 3 y 4 bits empezando la cuenta en posiciones diferentes al cero. Las líneas de datos generadas serán seccionados por el multiplexor.
- Un bloque Xilinx Counter configurado como contador libre de 1 bit, con el que se seleccionará una de las dos líneas de datos. Nótese que 1 es el número óptimo de bits necesarios para direccionar 2 líneas de datos.
- Un bloque Xilinx Mux configurado con 2 líneas de entrada y empleando la máxima precisión en los datos de salida.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

El sistema muestra a la salida las líneas de entrada de forma alterna. La salida tiene la máxima precisión necesaria, que es el máximo de las dos líneas de entrada, 4 bits.

La configuración puede observarse en las figuras 90 , 91 y 92.

Configuración

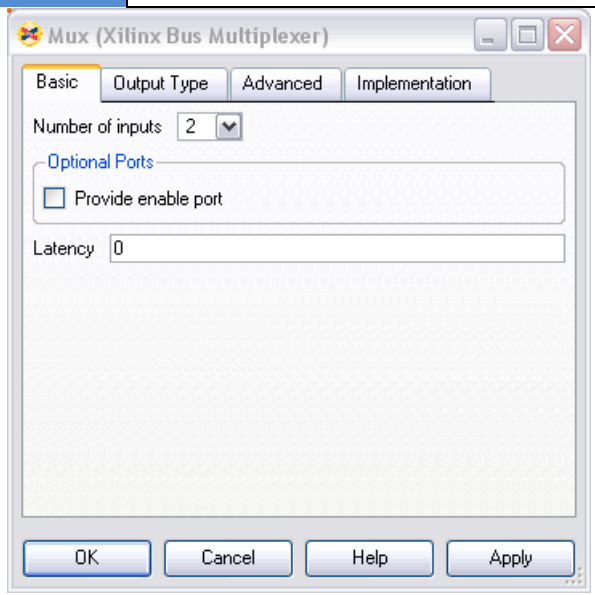

Figura 90. Configuración básica para el ejemplo funcionamiento del bloque Xilinx Mux

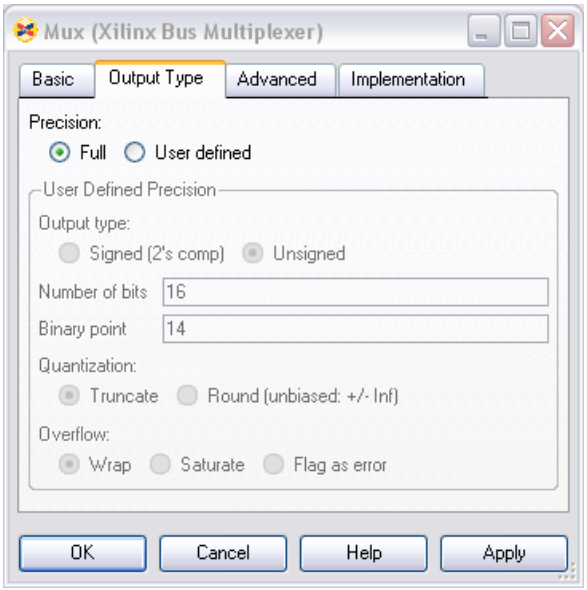

Figura 91. Configuración del tipo de salida para el ejemplo funcionamiento del bloque Xilinx Mux

Señales

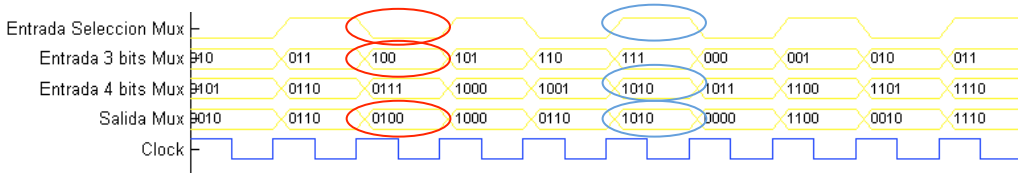

Figura 92.Señales de Ejemplo funcionamiento del bloque Xilinx Mux

3.4.16 Bloque Xilinx FFT v1_0

Descripción

El Bloque Xilinx FFT v1_0 (figura 93) implementa una Transformada Discreta de Fourier mediante el algoritmo Cooley-Turkey tipo radix-4.

El bloque opera con datos complejos.

Figura

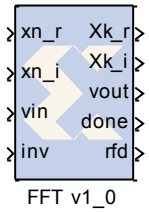

Figura 93. Bloque Xilinx CMult

Puertos de entrada y salida

Entrada

E-1 ***Xn_r***

Parte real del flujo de datos a la entrada

E-2 ***Xn_i***

Parte imaginaria del flujo de datos a la entrada

E-3 ***Vin***

Señal de validación de entrada. Marca los datos de entrada como válidos o inválidos.

E-4 ***Inv***

Señal de selección del comportamiento del bloque. Si la señal es cero, el bloque implementará una FFT, en caso de ser 1, el bloque implementará una IFFT.

Salida

E-1 ***Xk_r***

Parte real del flujo de datos a la salida

E-2 ***Xk_i***

Parte imaginaria del flujo de datos a la salida

E-3 ***Vout***

Validación de los datos de salida. Marca los datos de salida como válidos. Si la señal de validación marca una trama como invalida, la trama de salida correspondiente será marcada como invalida. Nótese que dados los retardos

internos del bloque, esto sucede para instantes de tiempo distintos.

E-4 ***Done*** - Hecho

Señala con un nivel lógico alto la primera muestra de una trama de salida

E-5 ***Rfd (Ready for data)*** – Preparado para datos

Indica con un nivel lógico alto que el bloque está preparado para recibir datos a la entrada

Configuración Básica

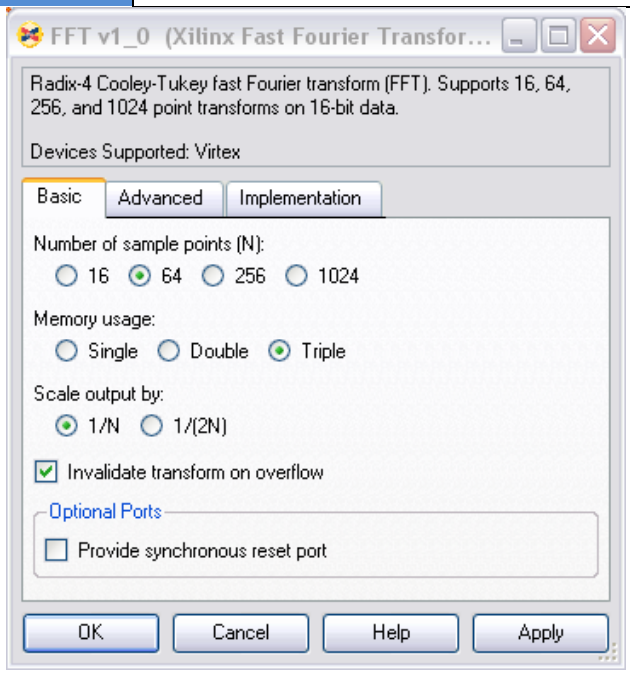

Figura 94. Ventana de Configuración básica del bloque Xilinx FFT v1_0 (primera pestaña)

Parámetros

1 ***Number of sample points*** – Número de puntos de muestreo

Tamaño de la Transformada Rápida de Fourier a implementar: `16, 46, 256` ò `1024`.

2 ***Memory Usage*** – Uso de memoria

Nivel de uso de los bancos de memoria para realizar la transformada: `simple, doble` o `triple`.

3 ***Scale Output by*** - Escalar salida por

Valor de escalado de los datos de salida: `1/N` o `1/(2N)`. Esta característica se explicará en detalle en las observaciones.

4	***Invalidate transform on overflow*** - invalidar transformada en caso de desbordamiento Configura el comportamiento del bloque en caso de desbordamiento del cálculo. Permite invalidar la salida o para la simulación.

Observaciones

- Este bloque es soportado únicamente por los dispositivos Virtex, existiendo el bloque Xilinx FFT v3_1 para el resto de familias de dispositivos.
- Puede emplearse la entrada de Validación, Vin, para evitar que el bloque procese datos que no contienen información, ya sea debido a retardos en bloques anteriores (emisor) o por encontrarse a la escucha el sistema (receptor)
- Existe un puerto opcional de reinicio, *`synchronous reset port`*. Ante un nivel alto en este puerto, el bloque vuelve a su estado inicial.
- El bloque permite emplear un periodo de muestreo distinto al detectado a la entrada.
- El bloque FFT/IFFT altera de forma considerable la forma del flujo de datos, debido a reescaldados numéricos. Como se ha explicado, entre los parámetros de configuración se puede elegir el reescalado de valor de los datos, entre 1/N y 1/(2N). Siendo n el exponente binario (2^n=N) necesario para obtener el tamaño de la transformada (16, 64, 256, 1024), n puede tomar los valores 4, 6, 8, y 10. Esto obliga a adaptar los datos a la entrada del bloque para que puedan soportar este reescalado sin perder información. Así, por ejemplo, en caso de emplearse un reescalado 1/N, será necesario añadir 4, 6, 8 y 10 posiciones decimales, respectivamente.
- Así mismo, el bloque FFT v1_0, altera la forma del flujo temporal, alterando el periodo de muestreo. El bloque introduce, en primer lugar, un retardo equivalente a un tiempo de trama completo (denominó tiempo de trama a tiempo equivalente al producto de la longitud de la trama a procesar por la FFT, 16, 64, 256 o 1024 muestras, según se defina, por el tiempo de muestreo de cada muestra), dado que tiene que esperar a la recepción completa de la trama antes de comenzar a procesar. Adicionalmente, emplea 2/3 de tiempo de trama en el procesado de la información, empleando el tercio de trama restante para entregar los datos a la salida (a una tasa triple, por tanto, a la de entrada, a este efecto lo denominaremos en adelante como de “ruptura de flujo”). Por tanto, emplea dos tiempos de trama desde que el primer dato de una trama determinada está disponible a la entrada, hasta que el último dato procesado correspondiente a dicha entrada está disponible a la salida. Esto hace necesario emplear un adaptador a la salida que devuelva

el flujo a su forma temporal anterior. El proceso se muestra gráficamente en la siguiente figura (figura 95):

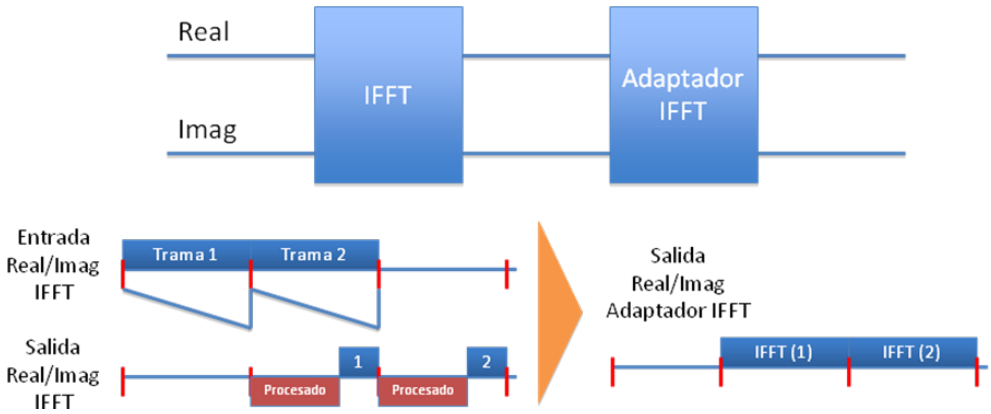

Figura 95. Representación gráfica de los flujos de entrada y salida del bloque Xilinx FFT v1_0, funcionado como IFFT y resultado del proceso de adaptación

- En el caso del emisor, como se explicará en el apartado correspondiente, el bloque se emplea en modo IFFT, por lo que el comportamiento es el mostrado en la figura __.
- En el caso del receptor, el bloque se emplea como FFT, para invertir el resultado de la IFFT. En este caso, no es necesario adaptar los datos a la entrada. Será necesario, sin embargo, escalar por N ó 2N (según sea el caso) la salida. Así mismo, también se constata la alteración del tiempo de muestreo explicado anteriormente. Adicionalmente, el flujo de datos a la salida de la FFT está invertido con respecto a la trama original. Esto es el primer valor de una trama a la salida de la FFT (receptor) inversora de una IFFT (emisor) se corresponde con el último valor de la trama original a la entrada de la IFFT. Por tanto, será igualmente necesario un bloque de post-adaptación que tenga en cuenta estos tres fenómenos. Su funcionamiento se explica gráficamente en la siguiente figura (figura 96):

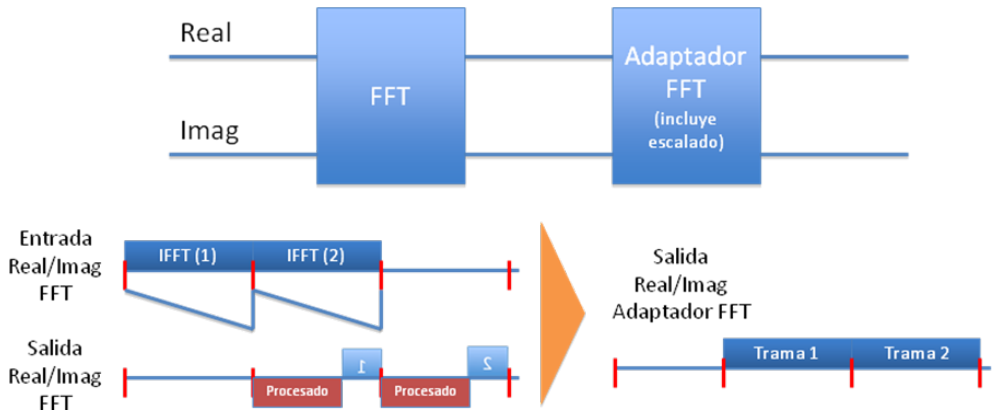

Figura 96. Representación gráfica de los flujos de entrada y salida del bloque Xilinx FFT v1_0, funcionado como FFT y resultado del proceso de adaptación

Ejemplo de Funcionamiento

Modelo

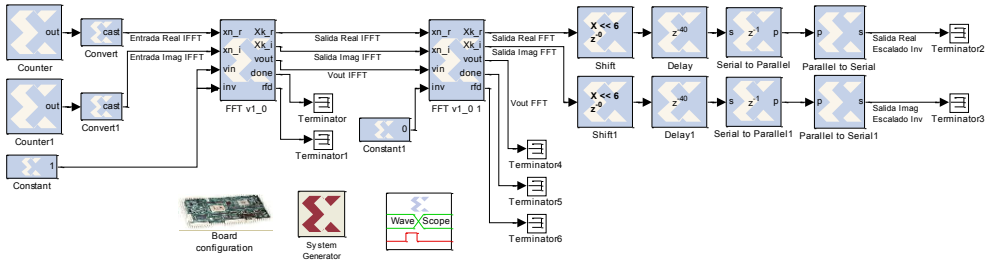

Figura 97. Modelo de ejemplo de funcionamiento del bloque Xilinx FFT v1_0

Descripción

Este modelo consta de:

- Dos fuentes de datos para la que se emplean dos bloques Xilinx Counter configurados como contadores libres de 4 bits con signo. Estas fuentes alimentan las líneas de datos real e imaginaria respectivamente.
- Dos bloques Xilinx Convert que añaden seis posiciones decimales, para poder entrar al bloque FFT y no perder información. En este caso se añaden 6 posiciones decimales porque se emplea una FFT de N=64 (n=6).
- Dos bloques Xilinx FFT v1_0 consecutivos. El primero configurado como FFT y el segundo como IFFT. Se han definido con N=64 y reescalado 1/N.
- Dos bloques Xilinx Shift para deshacer el reescalado. Se realiza un desplazamiento lógico a la izquierda 6 posiciones. Las posiciones decimales se eliminan directamente definiendo el tipo de salida con 4 bits sin punto binario.
- Dos bloques Xilinx Delay, dos bloques Xilinx Serial to Paralell y dos bloques Xilinx Paralell to serial. La conjunción de los bloques serie a paralelo y paralelo a serie se emplean para deshacer la inversión de orden. El bloque de retardo se emplea para alinear la trama al instante adecuado.
- Un bloque Xilinx WaveScope para observar la forma de las señales a la entrada y salida del bloque.

El modelo muestra como se recompone la señal de entrada y el efecto de “ruptura de flujo” comentado anteriormente y cuya solución se mostrará en el presente proyecto.

La configuración puede observarse en las figuras 98 , 99, 100, 101 y 102.

Configuración

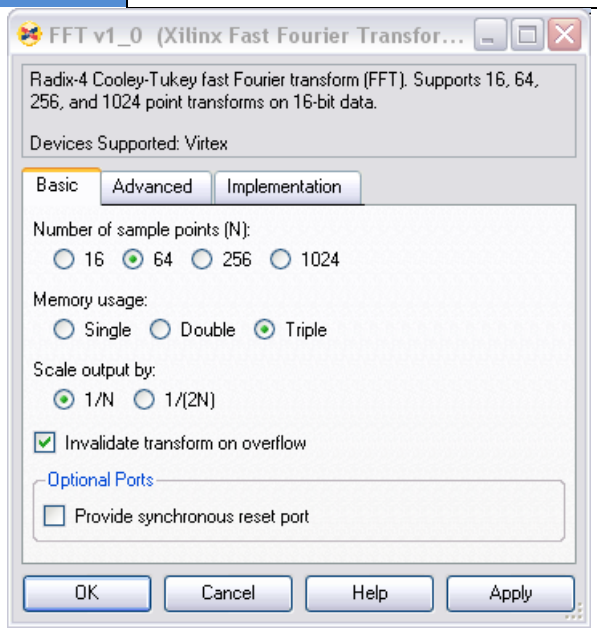

Figura 98. Configuración básica para el ejemplo funcionamiento del bloque Xilinx FFT v1_0, para el bloque IFFT

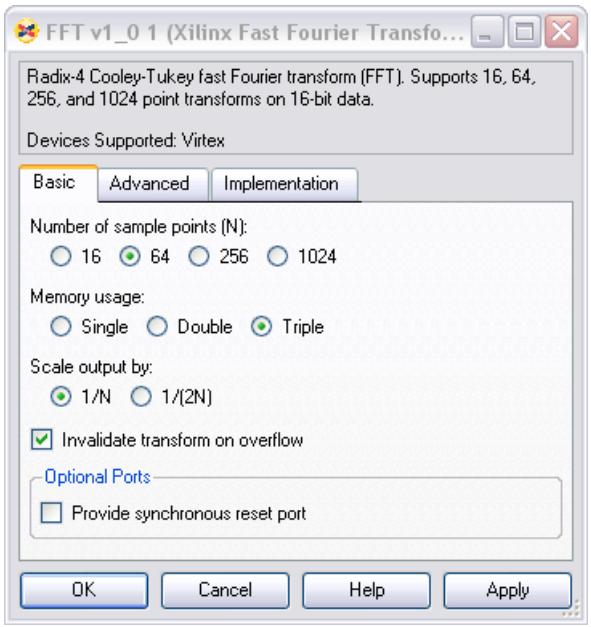

Figura 99. Configuración básica para el ejemplo funcionamiento del bloque Xilinx FFT v1_0, para el bloque FFT

Señales

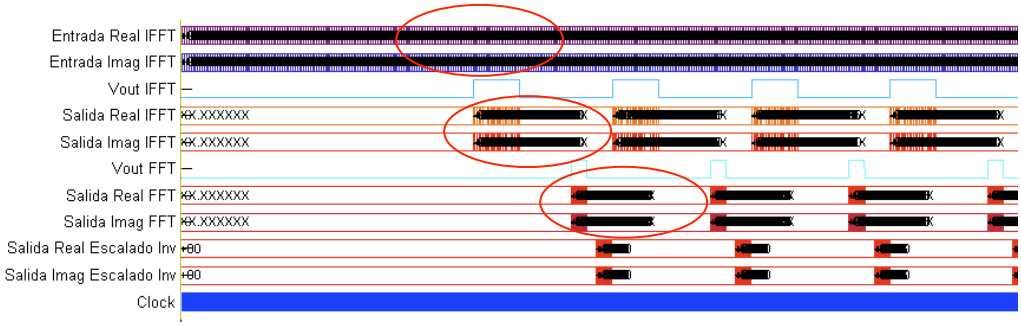

Figura 100.Señales de Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Ruptura del Flujo

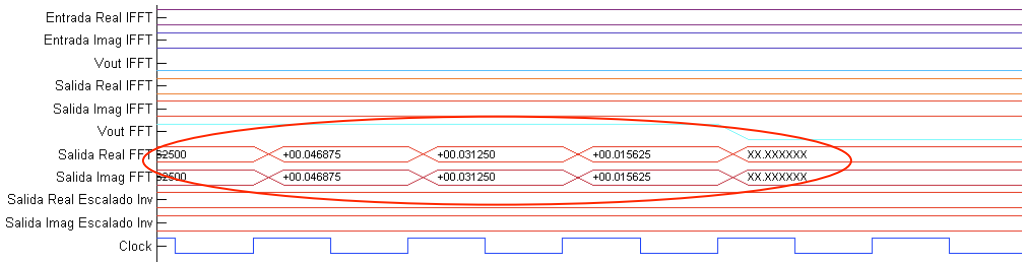

Figura 101.Señales de Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Reescalado e inversión de orden.

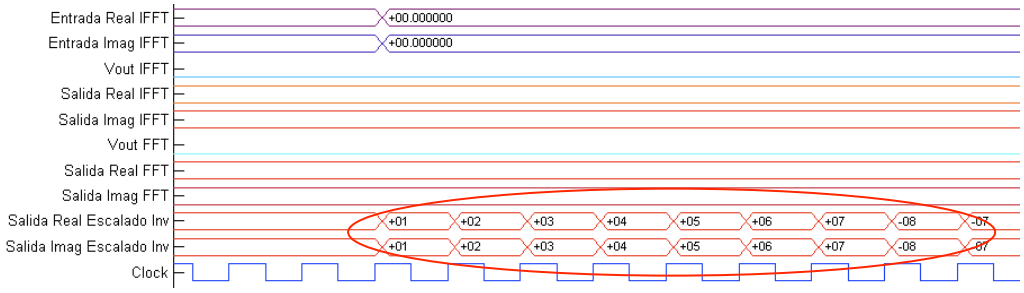

Figura 102.Señales de Ejemplo funcionamiento del bloque Xilinx FFT v1_0. Salida adaptada en escala y orden.

4. DISEÑO DEL EMISOR

El emisor propuesto en el presente proyecto se ha elaborado empleando bloques los Xilinx descritos en el apartado anterior. El emisor se ha diseñado para atender a los requisitos del estándar IEEE 802.16e-2005, empleando MIMO 2x2, OFDM en capa física y modulación de datos empleando BPSK. El diagrama de bloques correspondiente se muestra en la figura 103.

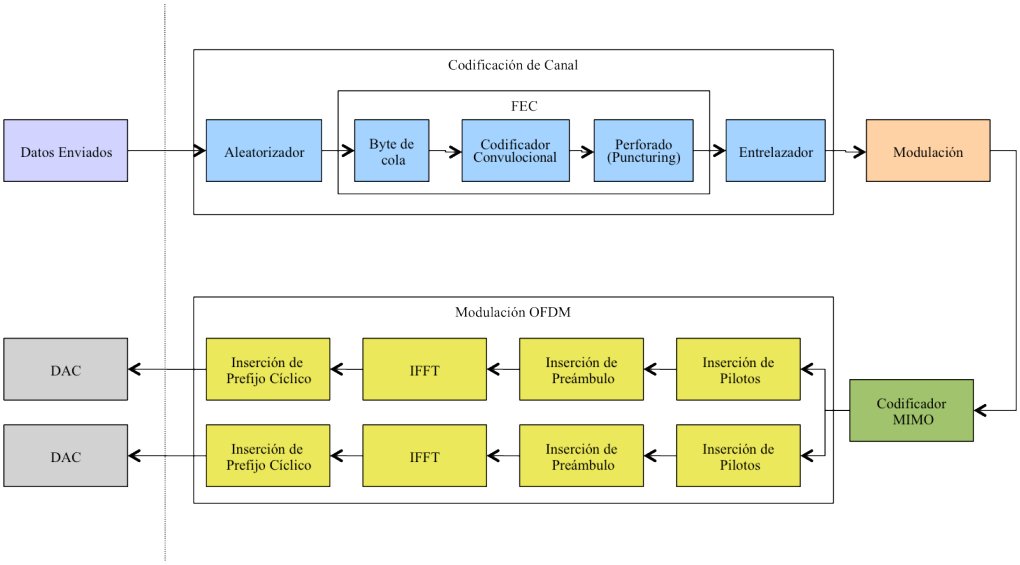

Figura 103. Descripción del emisor 802.16d-2004, MIMO 2x2, OFDM y BPSK

Para el diseño del emisor, se asumieron las siguientes simplificaciones:

- Se empleó un preámbulo simplificado (todo a uno), debido a que, para el trabajo conjunto emisor-receptor, se excluyó la existencia de canal y sus efectos, por lo que no es necesario sincronizar la trama en el receptor.
- Algunos elementos han sido adaptados para una estructura de trama conocida con anterioridad, formada por 10 símbolos OFDM, con modulación BPSK.

Así mismo, el modelo diseñado para demostración, posee la lógica adicional para la generación y comparación de las señales necesarias.

El diseño inicial del receptor se elaboró con el objetivo de soportar las distintas modulaciones de datos definidas en el estándar: BPSK, QPSK (Modulación en Cuadratura por Salto de Fase), 16-QAM y 64-QAM (Modulación por Amplitud en Cuadratura).

Debido a la complejidad asociada al manejo síncrono de las tramas, se decidió emplear modulación BPSK, si bien numerosos elementos del emisor están diseñados para manejar la complejidad de datos asociada a modulaciones más complejas.

Estas simplificaciones permiten, manteniendo los requisitos básicos recogidos en el estándar IEEE 802.16d-2004, reducir la complejidad del diseño.

En los apartados siguientes se describe cada uno de los bloques indicados en la figura 103, justificando su estructura y configuración para cumplir los requisitos del estándar IEEE 802.16d-2004.

4.1 Aleatorizador

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16d-2004 establece la necesidad de realizar una aleatorización de los bits de información con el objetivo de evitar largas secuencias de 0 y 1.

Para ello, se emplea un registro de desplazamiento de 15 bits con 2 funciones XOR y un vector de reinicio al inicio de cada trama ([1 0 0 1 0 1 0 1 0 0 0 0 0 0 0]), tal y como se muestra en la figura 104:

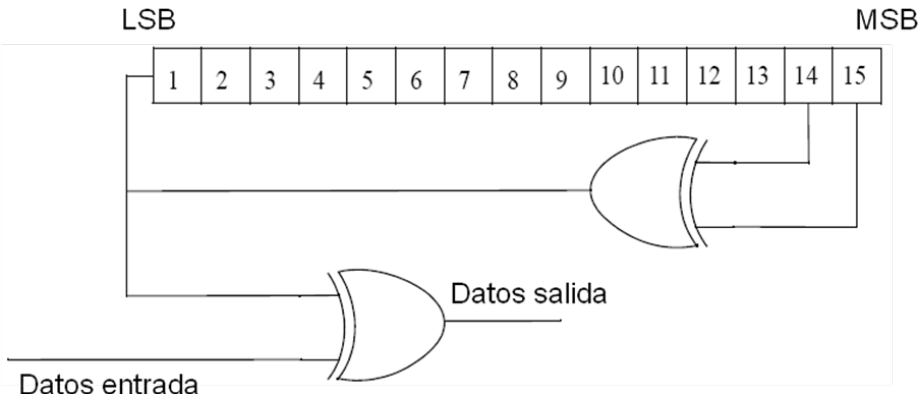

Figura 104. *Aleatorizador IEEE 802.16d-2004. Fuente: IEEE 802.16d-2004 [2]*

Por tanto, el receptor deberá contar con una etapa que realice la operación inversa con el fin de recuperar los bits de información transmitidos originalmente. En este caso, el Desaleatorizador del receptor se define de la misma manera que el Aleatorizador del emisor.

b) Diseño y Justificación

La aleatorización de los datos se realiza vía software mediante un bloque Mcode, tal y como se muestra en la figura 105.

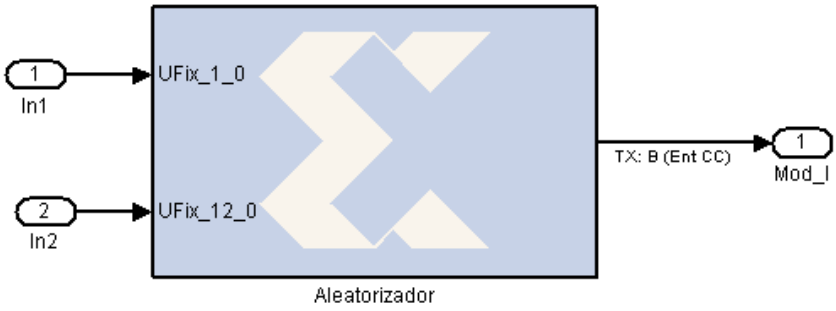

Figura 105. Modelo diseñado para el bloque de aleatorización

La configuración del bloque se muestra en la figura 106 y el código empleado puede consultarse en el apartado Anexo B de la presente memoria.

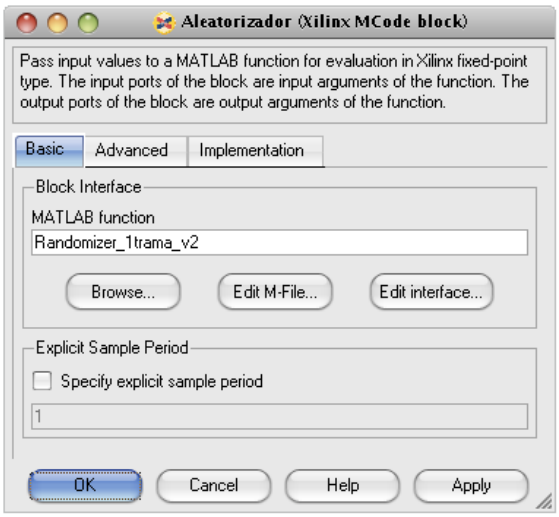

Figura 106. Configuración del Bloque Mcode para aleatorización.

4.2 Codificador Convolucional

a) Requisitos IEEE 802.16d-2004

Tras la aleatorización de los bits de información, se emplea un Codificador Convolucional con un código nativo de tasa 1/2, longitud restringida de 7 y dos polinomios generadores:

- Salida X : $G_1=171_{OCT}$
- Salida Y: $G_2=133_{OCT}$

La figura 107 muestra la representación esquemática del Codificador Convolucional.

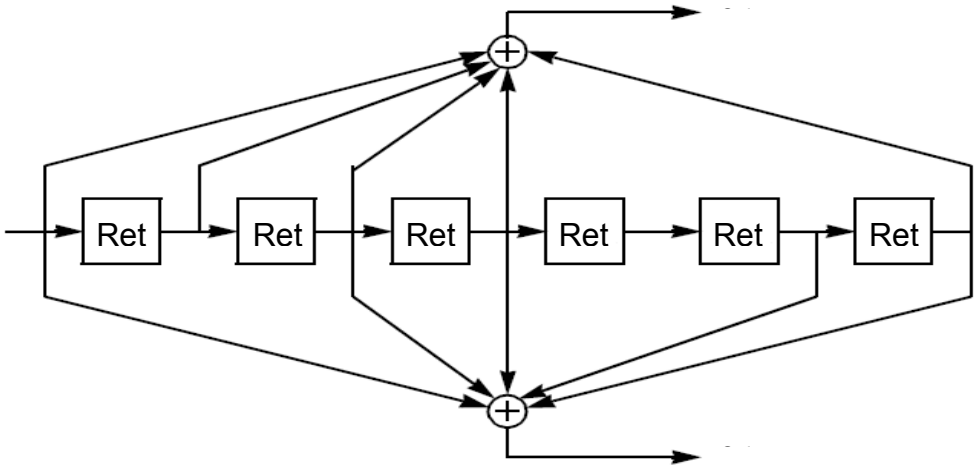

Figura 107. *Codificador Convolucional IEEE 802.16d-2004. Fuente: IEEE 802.16d-2004 [2]*

La salida de este bloque está formada, por tanto, por dos líneas de datos (X e Y)

b) Diseño y Justificación

Tras y como se muestra en la figura 108, como paso previo al codificador convolucional se introduce el tailbyte, en caso necesario. Para ello se emplea un bloque Xilinx ® MCode.

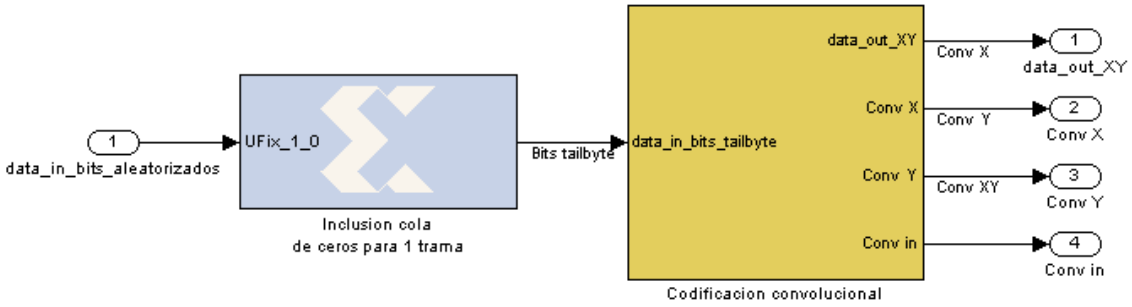

Figura 108. Codificador Convolucional con Inserción de Tailbyte previa.

Para realizar la operación de codificación convolucional se emplea un bloque Xilinx ® específico denominado *Xilinx ® Convolutional Encoder*. El conexionado del bloque se muestra en la figura 109. En dicha figura puede observarse como se emplean dos bloques *Xilinx ® Constant* con valores a 1, para el puerto vin (validación de la entrada) y 0, para el puerto rst (*reset*).

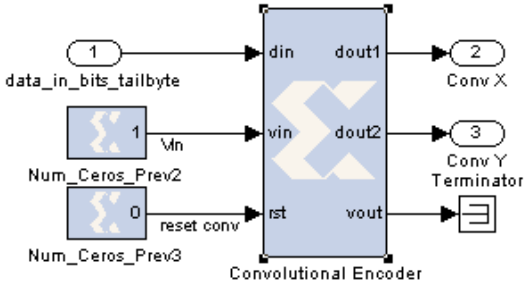

Figura 109. Modelo diseñado para el Codificador Convolucional.

La configuración del bloque específico se muestra en la figura 110, en ella se muestra como el bloque se configura con los códigos convolucionales que se desean emplear, tantos como salidas se deseen, en formato octal. En este caso, tal y como se ha descrito en el apartado de requisitos, dichos códigos son (171) y (133). El parámetro *constraint length* se configura con la longitud restringida (número de retardos más uno), 7 en este caso.

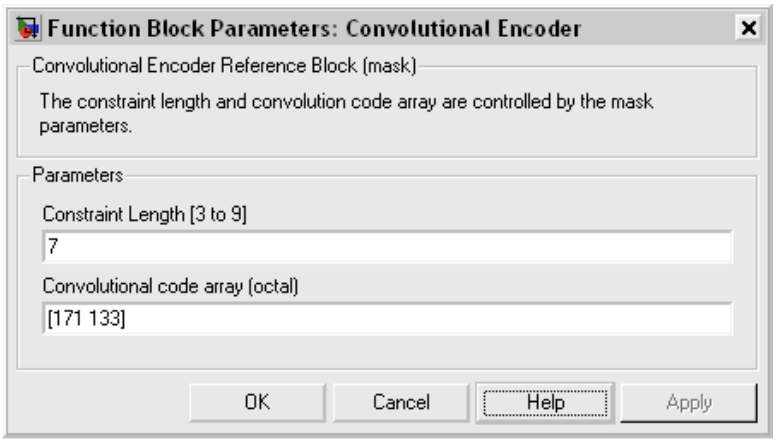

Figura 110. Configuración del bloque Xilinx ® específico para codificación convolucional.

4.3 Perforador (puncturer)

a) Requisitos IEEE 802.16d-2004

El proceso de perforación (*puncturing*) tiene el objetivo de ajustar la tasa de transmisión. Para el caso de BPSK, al emplear una tasa de codificación 1/2, el proceso de perforación consiste en concatenar las salidas del Codificador Convolucional ($X_1,Y_1 \rightarrow X_1Y_1$).

b) Diseño y Justificación

La operación de perforación se realiza, tal y como muestra la figura 111, un bloque *Xilinx ® Concat.* Este bloque concatena la líneas X e Y provenientes del bloque anterior y entregan una sola salida con los datos concatenados. Su configuración se muestra en la figura 112.

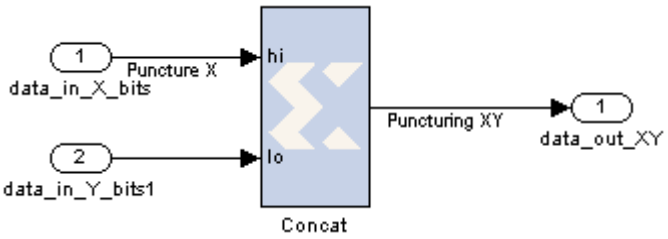

Figura 111. Modelo diseñado para el Perforador

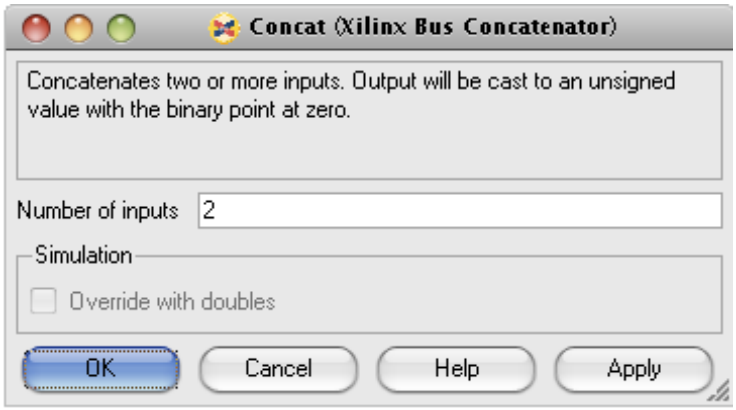

Figura 112. Configuración del bloque Xilinx ® Concat para el Perforador

4.4 Entrelazador

a) Requisitos IEEE 802.16d-2004

El entrelazado (interleaving) tiene un doble objetivo:

- Asegurar que los bits codificados consecutivamente no vayan a subportadoras adyacentes.
- Evitar que se formen colas de bits menos significativos al transformar los bits codificados en puntos de la constelación.

El proceso de entrelazado se realiza en dos pasos. Las expresiones que definen este proceso de entrelazado, las siguientes:

- Primera permutación: mk = (Ncbps/12) · kmod12 + floor(k/12)
- Segunda permutación: $j_k = s \cdot floor(m_k/s) + (m_k + N_{cbps} - floor(12 \cdot m_k/N_{cbps}))_{mod(s)}$

Donde*:*

- k es la posición de un bit codificado antes de la primera permutación (k = 0, 1, …, Ncbps-1).
- mk es la posición del bit k tras la primera permutación y antes de la segunda.
- jk es la posición del bit k tras la segunda permutación.
- Ncbps es el número de bits codificados en un bloque.
- Ncpc es el número de bits codificados por subportadora (1, 2, 4 o 6 para BPSK, QPSK, 16-QAM o 64-QAM, respectivamente).
- s = ceil(Ncpc/2).

Respecto al tamaño de bloque empleado, el estándar IEEE 802.16e-2005 establece un tamaño de 192 bits en el caso de emplear modulación BPSK.

b) Diseño y Justificación

El diseño del bloque entrelazador se muestra en la figura 113. La operación es realizada en un bloque Xilinx ® Bitbasher que reordena el flujo de datos, para lo cual opera sobre bloques de 192 bits (96 símbolos de 2 bits). La configuración de este bloque se muestra en la figura 113.

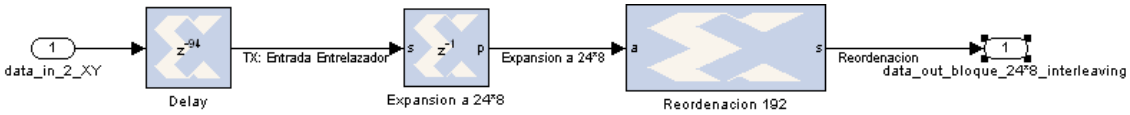

Figura 113. Modelo diseñado para el Entrelazador

Los bloques previos se emplean para preparar el bloque de datos de forma adecuada. Este acondicionamiento lo realiza un bloque de retardo que alinea la ventana. Este retardo es necesario debido a los retardos introducidos bloques necesarios y se realiza para trasladar el bloque de datos a la siguiente ventana de operación del siguiente bloque, *Xilinx ® Serial to Parallel* (ver figura 114), quien prepara el bloque de 192 bits y lo pone en paralelo, para su posterior reordenación (ver figura 115).

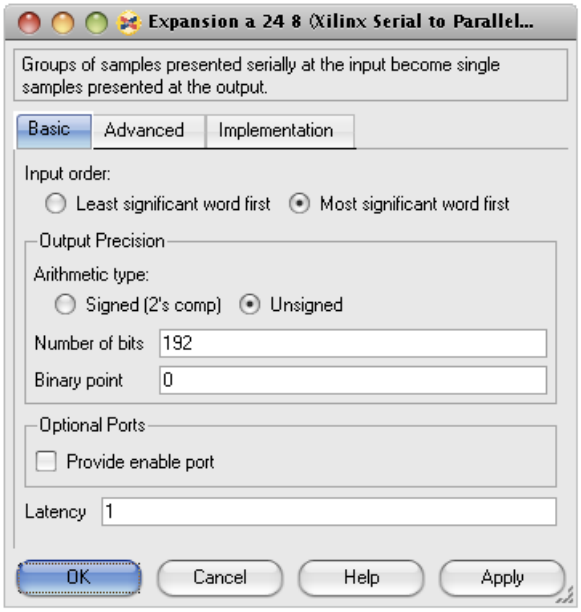

Figura 114. Configuración del bloque Xilinx ® Serial to Parallel para el Entralazador

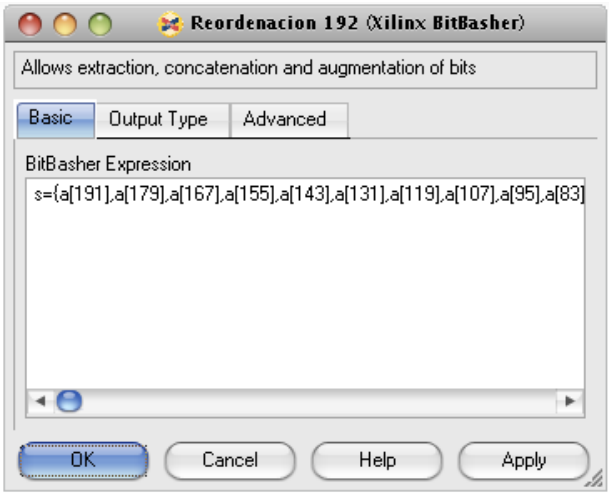

Figura 115. Configuración del bloque Xilinx ® Bitbasher para el Puncturer

4.5 Modulador

a) Requisitos IEEE 802.16d-2004

La etapa de modulación soporta constelaciones del tipo BPSK, QPSK, 16-QAM y 64-QAM. Además, cada punto de la constelación (donde b_0 indica el bit menos significativo) debe ser normalizado mediante un factor 'c' para conseguir la misma potencia media, como se muestra en la figura 116:

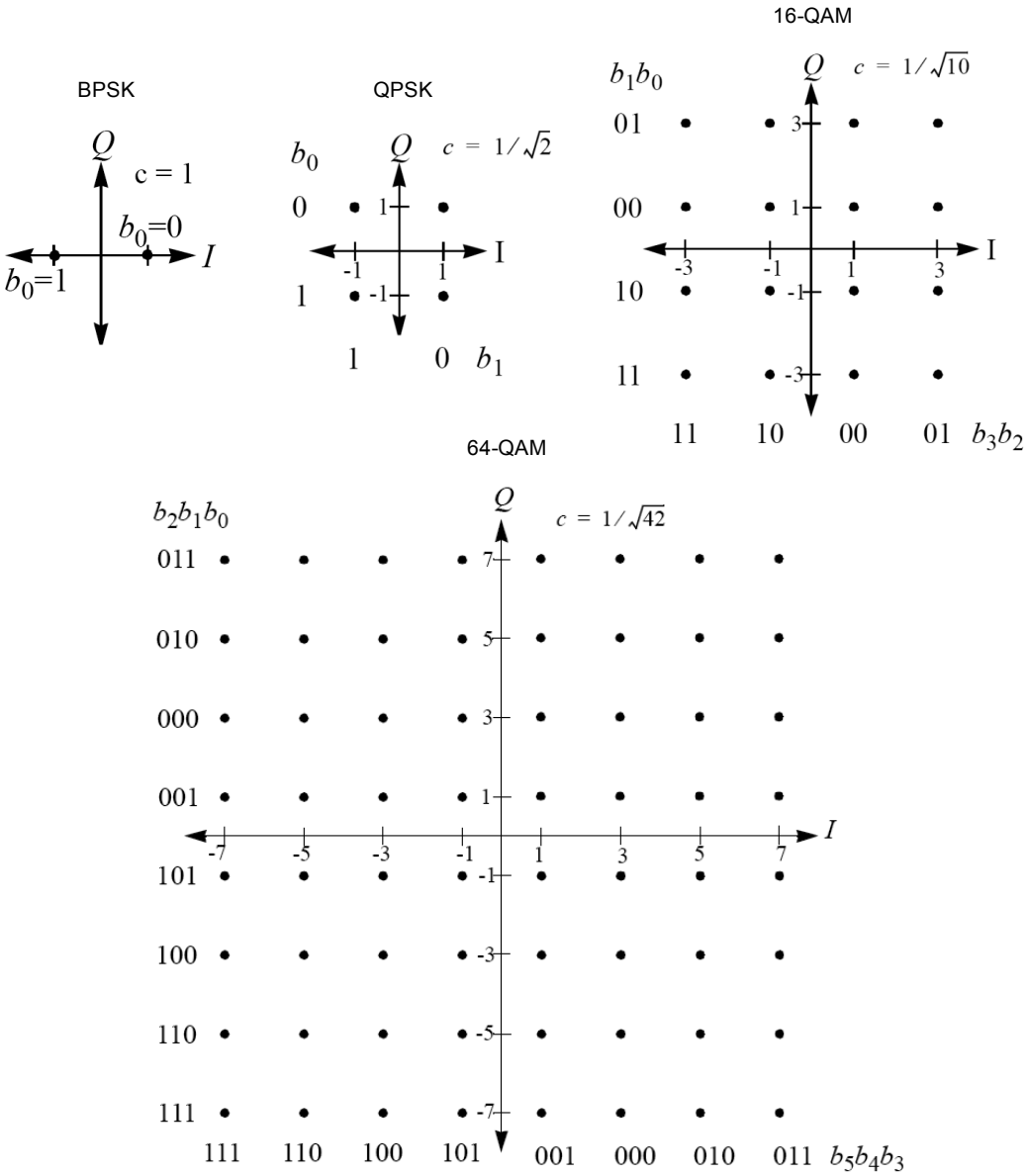

Figura 116. *Constelaciones BPSK, QPSK, 16-QAM y 64-QAM IEEE 802.16d-2004. Fuente: IEEE 802.16d-2004 [2]*

b) Diseño y Justificación

El modelo implementado en este prototipo realiza un modulación BPSK, esto es, tal y como semana la figura 117, asignar los valores 1 y -1 a los valores binarios de entrada 0 y 1 respectivamente.

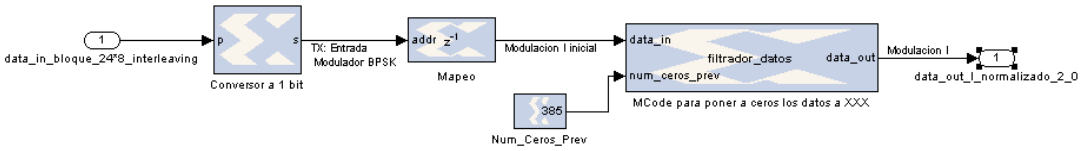

Figura 117. Modelo diseñado para el Modulador

Para ello se emplea una memoria ROM configurada, tal y como se muestra en la figura 118, con un vector de inicialización [1, -1], mediante el que se indica el vector de conversión. Los valores [0,1] se convierten en [1,-1]. Dado que son necesarios 2 bits para representar estos valores, se fija una profundidad (*Depth*) de 2.

Al emplearse en este caso una codificación BPSK, solo es necesario codificar la línea I. Para codificaciones superiores se debería emplear una memoria ROM por cada una de las líneas (I y Q).

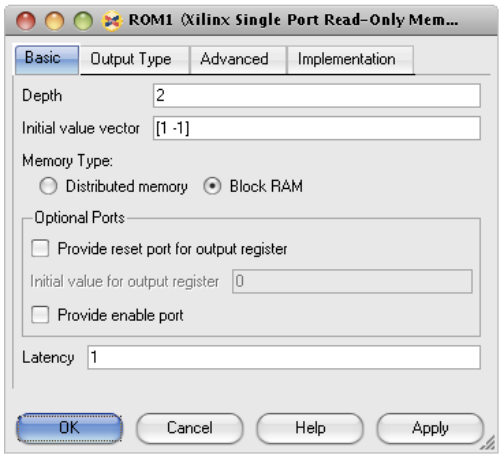

Figura 118. Configuración del bloque Xilinx ® ROM para el Modulador

Previo a la realización de esta operación, es necesario serializar el flujo proveniente del bloque anterior, para que los datos entren al modulador de bit en bit. Esta operación la realiza un bloque *Xilinx ® Parallel to Serial* configurado tal y como se muestra en la figura 119.

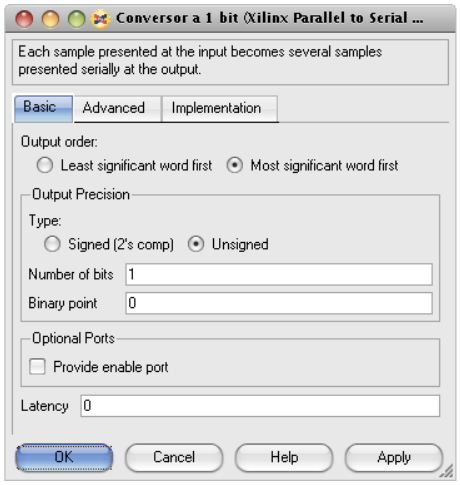

Figura 119. Configuración del bloque Xilinx ® Parallel to Serial para el Modulador

4.6 Codificador MIMO

a) Requisitos IEEE 802.16d-2004

Para incrementar el radio de la celda cubierta y aumentar la capacidad de la comunicación, el estándar IEEE 802.16e-2005 ofrece a posibilidad de emplear técnicas MIMO. Se aplicará esta técnica de diversidad empleando dos antenas. Para ello se realiza la transmisión de la siguiente manera:

- Primer uso del canal: la antena 0 transmite s0 y la antena 1 transmite s1.
- Segundo uso del canal: *la antena 0 transmite* $-s_1^*$ *y la antena 1 transmite* s_0^*.

Donde:

- s0 es el primer símbolo que llega al codificador MIMO.
- s1 es el segundo símbolo que llega al codificador MIMO.
- * representa el conjugado.

Debe indicarse que esta operación se realiza sobre cada elemento modulado, teniendo pleno sentido para el caso de emplearse señales en cuadratura.

b) Diseño y Justificación

El bloque de codificación MIMO opera sobre las líneas I y Q generando dos juegos separados de líneas I y Q, sobre las que trabajarán todos los bloques siguientes hasta llegar hasta la antena.

El conjunto del bloque se muestra en la figura 120.

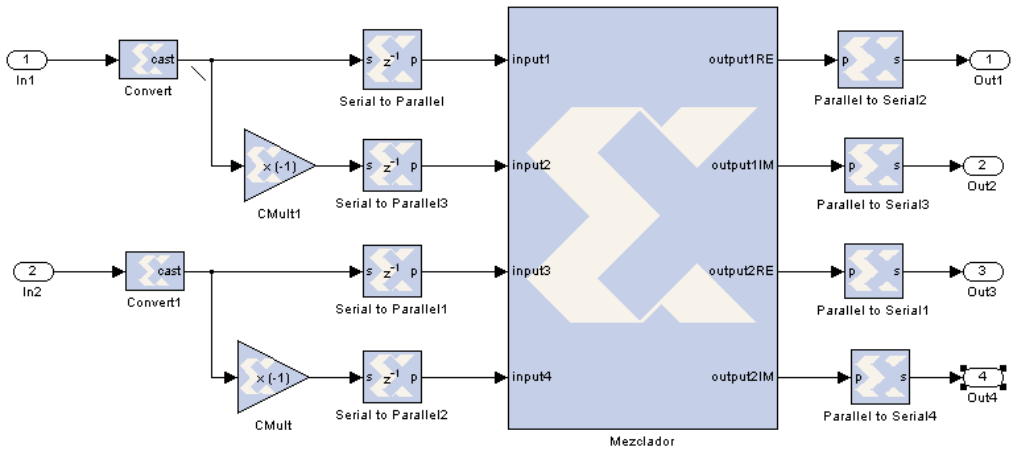

Figura 120. Modelo diseñado para el bloque de codificación de pilotos

El modelo está formado por cinco elementos diferenciados:

- *Bloque de adaptación del formato de datos*: debido a que el bloque requiere el manejo de negativos y conjugados, se emplearán dos bits para representar cada símbolo BPSK, y evitar el desbordamiento de los bloques de inversión (que se realiza en complemento a dos). Se emplea un bloque *Xilinx*$^{(R)}$ *Cast* (convert) para ello.

- *Bloques de inversión:* Se dividen las líneas I y Q en dos flujos distintos, uno de ellos se invierte y otro no se altera. De esta forma, ponemos a disposición de los bloques siguientes, muestras positivas y negativas de cada símbolo, para ser combinadas tal y como se indica anteriormente en los requisitos.

- *Bloques serie a paralelo:* este bloque toma dos símbolos BPSK (2 bits por símbolo) para poder realizar las operaciones que se describen en los requisitos.

- *Bloque mezclador:* este bloque tomas los negativos y los positivos de las líneas y Q del primer y segundo símbolo original (que han sido puesto en paralelo por el bloque anterior) para componer la codificación MIMO descrita en los requisitos. La configuración de este bloque se describe en la figura 121.

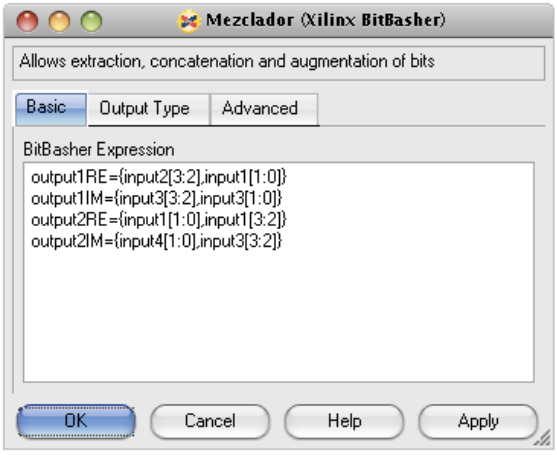

Figura 121. Configuración del bloque "mezclador" del codificador MIMO

Debe tenerse en cuenta que la salidas están definidas en función de los dos usos consecutivos. De este manera, por ejemplo, la Salida 1 ("output1RE y output1IM) toma s0 (real + j·imaginaria) y –s1* (-real + j·imaginaria). Para la salida real (output1RE) esto supone tomar de la entrada 1 (input1, parte real) los dos primeros bits (0 y 1) (parte real de s0) y de la entrada 2 (intput2, parte real negativa) los dos últimos bits (2 y 3) (parte real de S1, negativa). Para la salida Imaginaria (output1IM) esto supone tomar de la entrada 3 (input3, parte imaginaria) los dos primeros bits (0 y 1) (parte imaginaria de s0) y de la misma entrada 3 los dos últimos bits (2 y 3) (parte imaginaria de s1).

- *Bloque paralelo a serie:* el bloque reinyecta en serie la salidas conformadas, implementando los dos usos consecutivos del canal.

4.7 Inserción de pilotos

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16e-2005 establece la inserción para cada símbolo OFDM de una serie de subportadoras a las de datos ya existentes. Por tanto, el símbolo OFDM queda conformado de la siguiente forma (para el caso BPSK):

- Subportadora DC: 1 subportadora nula en el punto de frecuencia media. Esta subportadora marca el punto "0" de la trama, numerándose las subportadoras desde el -128 (menor frecuencia) al 127 (mayor frecuencia).
- Banda de guarda inferior: 27 subportadoras nulas, antes de las frecuencias más bajas (-128 a -101)
- Banda de guarda superior: 28 subportadoras nulas, tras las frecuencias mas altas (99 a 127)
- 8 pilotos. Se incluyen con el objetivo de estimar el canal en el receptor y realizar la compensación adecuada.
- 192 subportadoras de datos.

El esquema final de la trama, con 256 portadoras, es el que se muestra en la figura 122.

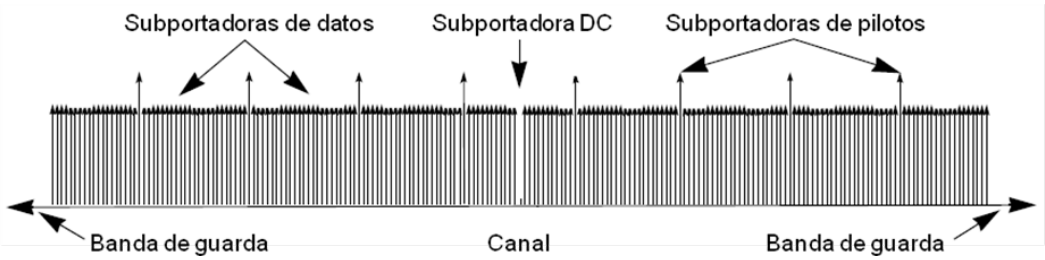

Figura 122. *Descripción en frecuencia del símbolo OFDM IEEE 802.16d-2004. Fuente: IEEE 802.16d-2004 [2]*

b) Diseño y Justificación

El bloque de inserción de preámbulo se diseña para cada una de las dos líneas de antena, realizando la misma operación de forma simultanea sobre las líneas I y Q de cada una de las líneas de antena.

El conjunto del bloque se muestra en la figura 123.

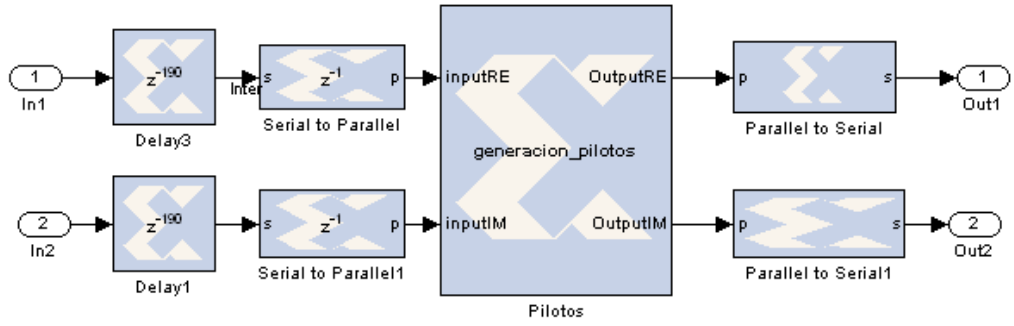

Figura 123. Modelo diseñado para el bloque de inserción de pilotos

El modelo está formado por cuatro elementos diferenciados:

- *Bloque de retardo*: este bloque retarda la señal para prepararla para las operaciones del siguiente bloque. En este caso, introduce un retardo de 190 muestras.

- *Bloque Serie a Paralelo:* este bloque prepara la trama de 192 símbolos BPSK, 384 bits (2 bits por símbolo). Cada uno de estos símbolos BPSK se constituye en una subportadora (como se has descrito anteriormente). Para ello, pone en paralelo la trama para que el siguiente bloque pueda trabajar sobre la trama al completo.

- *Bloque "generación de pilotos":* este bloque toma las 192 subportadoras y le añade el resto de elementos que constituyen el símbolo OFDM. La salida está constituida por 256 símbolos BPSK, como se describe en la figura 122, lo que supone 512 bits. El código empleado en este bloque puede consultarse en el Anexo B de la presente memoria.

- *Bloque Paralelo a Serie:* este bloque inyecta el símbolo OFDM a la línea de datos en serie en forma de símbolos BPSK (2 bits por símbolo)

4.8 Inserción del preámbulo

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16e-2005 establece la inclusión de un preámbulo con el objetivo de permitir la sincronización entre el emisor y el receptor, así como la estimación del canal. Dicho preámbulo consiste en uno o dos símbolos OFDM de características conocidas que se inserta al principio de la trama. En el caso del emisor descrito, como se ha comentado anteriormente, se emplea un símbolo OFDM con todas las portadoras con valor 1.

b) Diseño y Justificación

El bloque de inserción de preámbulo se diseña para cada una de las dos líneas de antena, realizando la mismo operación de forma simultanea sobre las líneas I y Q de cada una de las líneas de antena.

El conjunto del bloque se muestra en la figura 124.

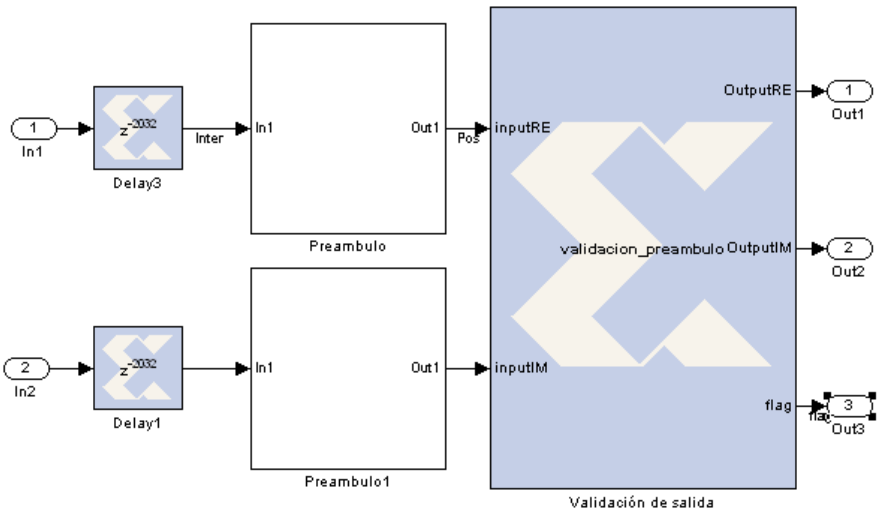

Figura 124. Modelo diseñado para el bloque de inserción de preámbulo

El modelo está formado por tres elementos diferenciados:

- *Bloque de retardo*: este bloque retarda la señal para prepararla para las operaciones del siguiente bloque. En este caso, introduce un retardo de 2032 muestras.

- *Bloque de Preámbulo:* este bloque realiza la operación de inserción propiamente dicho. El diseño y configuración de este bloque se explicará a continuación.

- *Bloque de Validación de salida*: este bloque actúa para mantener toda la línea de datos a cero a la espera del instante en que la entrada en dicho bloque debería contener datos válidos (debe notarse en que, dada la planificación estática del modelo sobre el que se está trabajando, dicho instante es conocido a priori). De esta forma, se evita que el siguiente bloque, IFFT, trabaje sobre datos no válidos, generando señales espurias hacia las antenas. Este bloque, por tanto, no es necesario para la realización de la inserción de preámbulos. Se trata de un bloque de adaptación de los flujos de la señal al diseño de nuestro modelo. El código empleado en este bloque puede consultarse en el Anexo B del presente documento.

El bloque de preámbulo emplea la configuración de "árbol de paralelización" (su funcionamiento se explica en el Anexo A de este documento) para ejecutar la operación de inserción sobre el total de la trama OFDM (10 símbolos OFDM). Debido al tamaño de la trama que se ha de manejar, se ha de realizar la paralelización en dos niveles. El diseño del primer nivel se muestra en la figura 125.

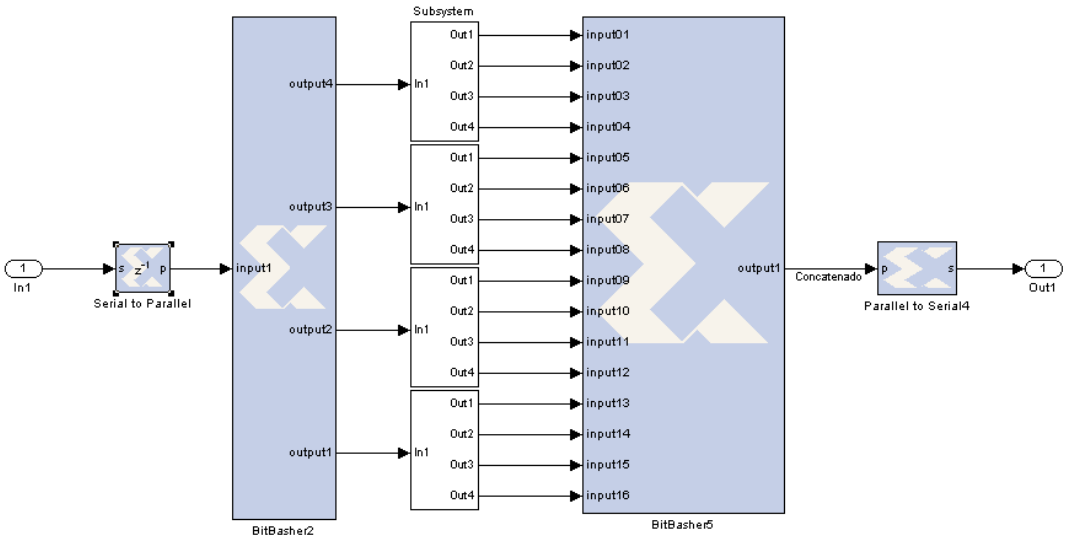

Figura 125. Bloque de Preámbulo. Primer nivel.

A este nivel se realiza una primera preparación. El bloque de operación del árbol es un subsistema completo que realiza la operación de inserción empleando un segundo "árbol de paralelización" (ver figura 127). Cabe señalar que la operación de

unificación se realiza en el primer nivel La configuración del bloque *Xilinx® Serial to Parallel* se muestra en la figura 126.

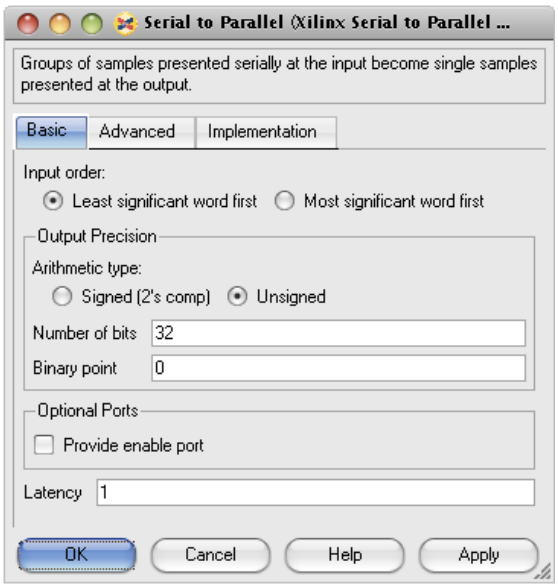

Figura 126. Configuración del Bloque Serie a Paralelo del "árbol de paralelización" del bloque de inserción de preámbulo (primer nivel)

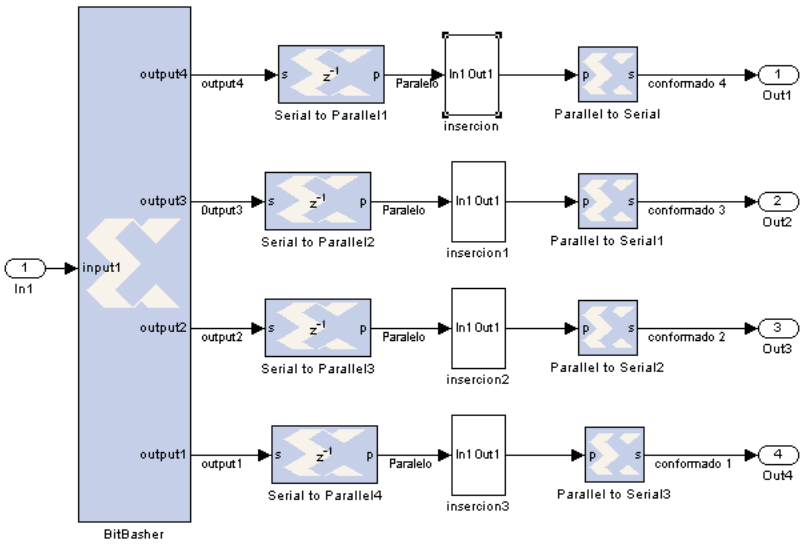

Figura 127. Modelo diseñado para el segundo nivel del bloque preámbulo.

Este segundo nivel realiza un proceso de paralelización adicional. Cabe destacar que el proceso de paralelización en dos niveles es necesario para modulaciones superiores a BPSK.

La configuración del bloque *Xilinx® Serial to Parallel* del segundo nivel se muestra en la figura 128.

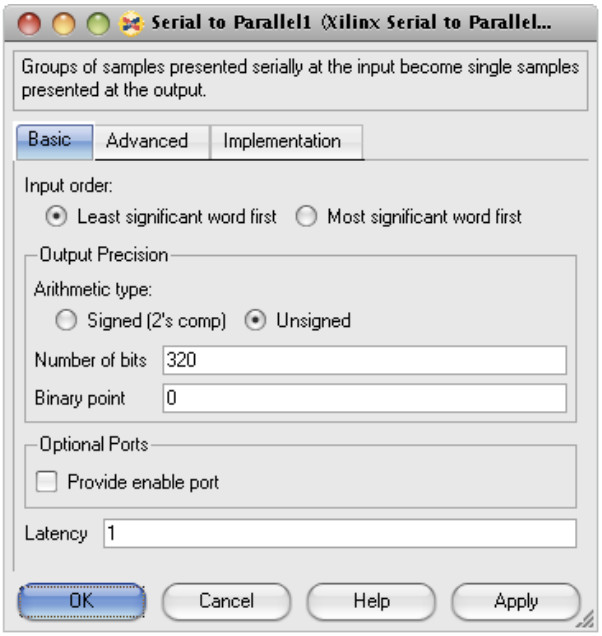

Figura 128. Configuración del Bloque Serie a Paralelo del "árbol de paralelización" del bloque de inserción de preámbulo (segundo nivel)

El bloque de operación del árbol realiza una concatenación de la señal entrante y la parte proporcional del preámbulo asignado a la porción paralela de la señal, tal y como se muestra en la figura 129.

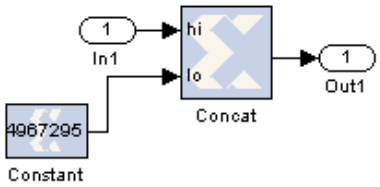

Figura 129. Modelo diseñado para la inserción de preámbulo.

En este punto, se están utilizando 2 bits para representar el símbolo BPSK. Cada símbolo OFDM está compuesto por 256 símbolos BPSK (incluidas guardas, pilotos, etc.). Por tanto, cada símbolo OFDM está compuesto por 2*256 = 512 bits. La trama total OFMD diseñada en este prototipo consta de 10 símbolos OFDM, 512*10 = 5120 bits. El sistema coloca toda la trama en 16 líneas en paralelo, manejando cada línea 5120/16 = 320 bits. El sistema consta de 4 bloques de preámbulo cada uno con 4 bloques de inserción (dos niveles). En total, el bloque de inserción de preámbulo consta de 16 bloques de inserción (uno para cada línea). El tamaño del preámbulo a insertar es de 256*2 = 512 bits (dos bits por símbolo BPSK). Cada bloque de

inserción es responsable de la inserción de 512/16 = 32 bits del preámbulo. Su salida está formada por 320 + 32 = 352 bits. La posterior fase de unificación de las líneas de datos obtienen 352*16 = 5632 bits, que se corresponden con 1+10=11 símbolos OFDM, 11*256*2= 5632 bits.

Cabe destacar que el modelo descrito en este apartado asume un preámbulo con 512 bits puestos a uno. Por tanto, todos los bloques de inserción son iguales, insertando 32 unos. En caso de diseñarse un preámbulo más complejo, se debería configurar cada bloque de inserción con la forma adecuada para obtener el preámbulo deseado.

4.9 IFFT

a) Requisitos IEEE 802.16d-2004

La operación IFFT crea la forma de onda temporal del símbolo OFDM. Esta operación se realiza una vez que dicho símbolo dispone en el dominio de la frecuencia de las subportadoras de datos, pilotos y nulas (bandas de guarda y subportadora de DC). La operación IFFT también se realiza sobre el preámbulo. Dado que la operación sobre el preámbulo se realiza sobre una trama de forma conocida, el resultado en forma temporal es también conocido.

Así, el tamaño de la IFFT viene determinado por el número de subportadoras que conforman el símbolo OFDM. En el caso de este proyecto (modulación BPSK), el tamaño de dicha IFFT es de 256 subportadoras.

Debe indicarse que en el receptor la operación de inversión (FFT) no se realiza sobre el preámbulo (dado su carácter conocido en ambos dominios). Como consecuencia, en el apartado de validación, este bloque deberá ser validado conjuntamente con el bloque de inserción de preámbulo.

b) Diseño y Justificación

La IFFT diseñada consta de dos sistemas idénticos, cada uno de los cuales realiza la operación sobre las líneas I y Q simultáneamente. Por tanto, existe un bloque IFFT por cada una de las antenas. El modelo diseñado es el que se presenta en la figura 130.

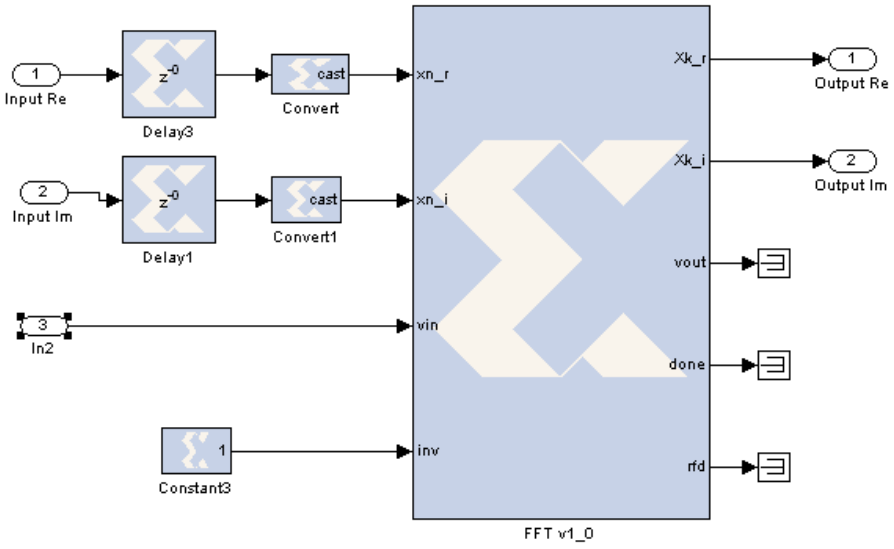

Figura 130. Modelo diseñado del bloque IFFT

La IFFT está configurada para realizar una operación de IFFT de 256 símbolos. Tal y como se describe en el apartado 3.5.16, el bloque Xilinx ® FFT v1_0, puede realizar un reescalado del orden 1/N o 1/2N. En este caso, se ha seleccionado el reescalado 1/N. Por tanto, la salida del bloque IFFT v1_0 es 256 veces menor. Para no perder información se ha de reformar el formato de datos. Un bloque Xilinx ® Cast "Convert" añade 8 posiciones decimales al símbolo BPSK con ese fin.

Tal y como se ha explicado anteriormente los bloques FFT e IFFT ocasionan una "ruptura del flujo" que ha de ser subsanado con posterioridad. Por razones de eficiencia, se aprovecha el siguiente bloque (Inserción de Prefijo Cíclico) para realizar dicha subsanación.

El bloque IFFT maneja un símbolo OFDM-BPSK de 256 símbolos de 10 bits cada uno. Debido al fenómeno de ruptura de flujo, el flujo de datos que este bloque entrega a l siguiente (Inserción de Prefijo Cíclico) está formado por 768 símbolos de 10 bits cada uno (a una tasa binaria triple).

La configuración de los bloques de conversión e IFFT se muestra en las siguientes figuras (figuras 131 y 132)

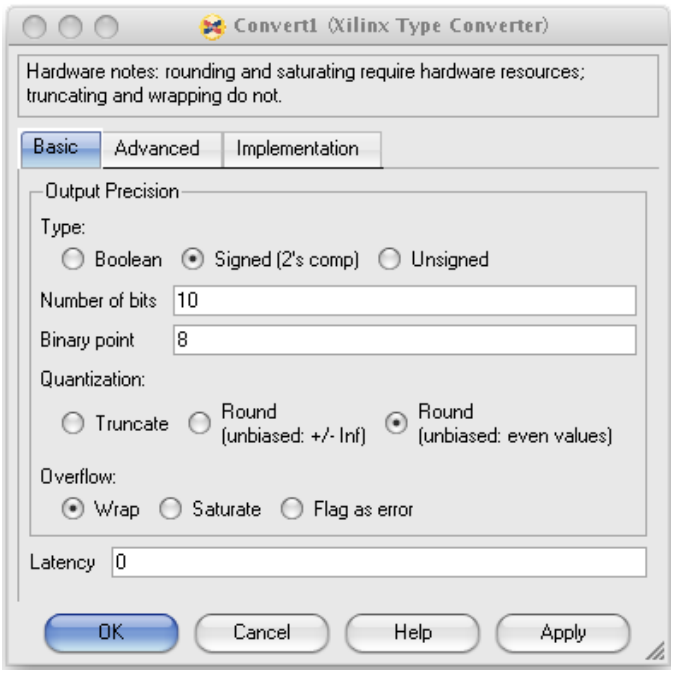

Figura 131. Configuración del bloque de conversión del formato de datos en el bloque IFFT

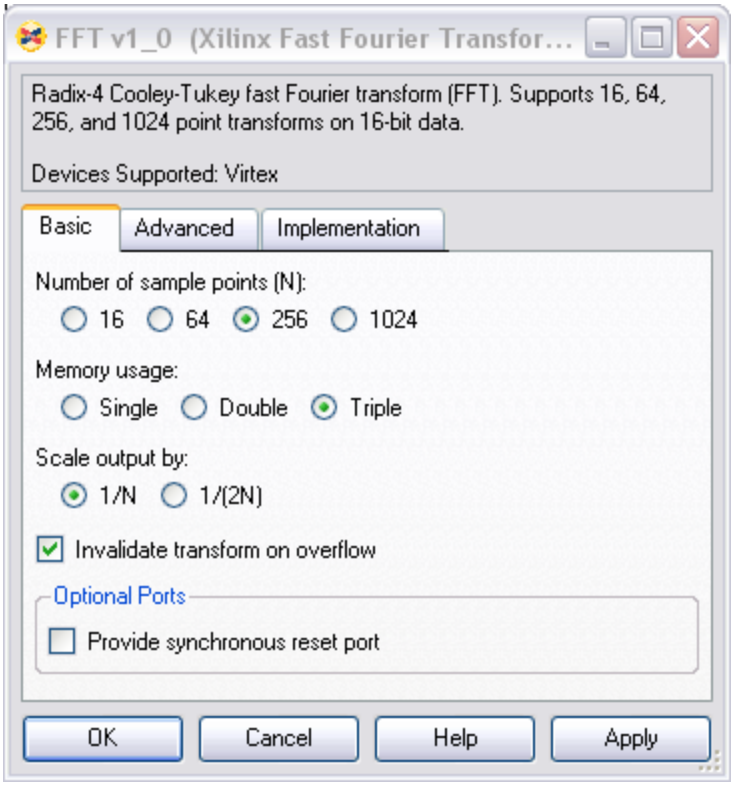

Figura 132. Configuración del bloque Xilinx IFFT en el bloque de IFFT

4.10 Inserción del prefijo cíclico

a) Requisitos IEEE 802.16d-2004

El estándar IEEE 802.16e-2005 establece la necesidad de copiar un número determinado de sus muestras finales al principio del mismo (tiempo T_g), que mantienen la ortogonalidad entre portadoras. Este proceso se muestra en la figura 133:

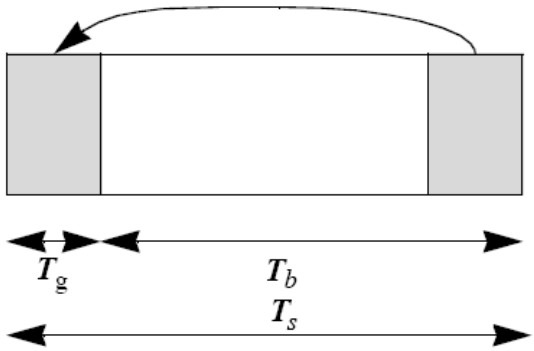

Figura 133. *Prefijo cíclico IEEE 802.16d-2004. Fuente: IEEE 802.16d-2004 [2]*

Dichas muestras se denominan CP (Prefijo Cíclico) y su objetivo es evitar la ISI (Interferencia entre Símbolos), principalmente debida a los efectos del multitrayecto.

El estándar IEEE 802.16e-2005 establece diferentes tamaños seleccionables para el prefijo cíclico, esto es, $G = T_g / T_b = 1/4, 1/8, 1/16$ o $1/32$.

La elección de un determinado tamaño de prefijo cíclico dependerá de la respuesta al impulso del canal, debiendo ser T_g mayor que ésta para que ningún rayo reflejado de un símbolo interfiera con el siguiente.

En este caso, se ha escogido $G = 1/8$, con lo que T_g=32. Las 32 ultimas portadoras temporales se replican al comienzo de la trama.

b) Diseño y Justificación

La Inserción del Prefijo Cíclico se realiza mediante cuatro bloques idénticos, que operan sobre cada una de las líneas I/Q de cada antena respectivamente. Para cada uno de esos bloques, el diseño es el que se muestra en la figura 134.

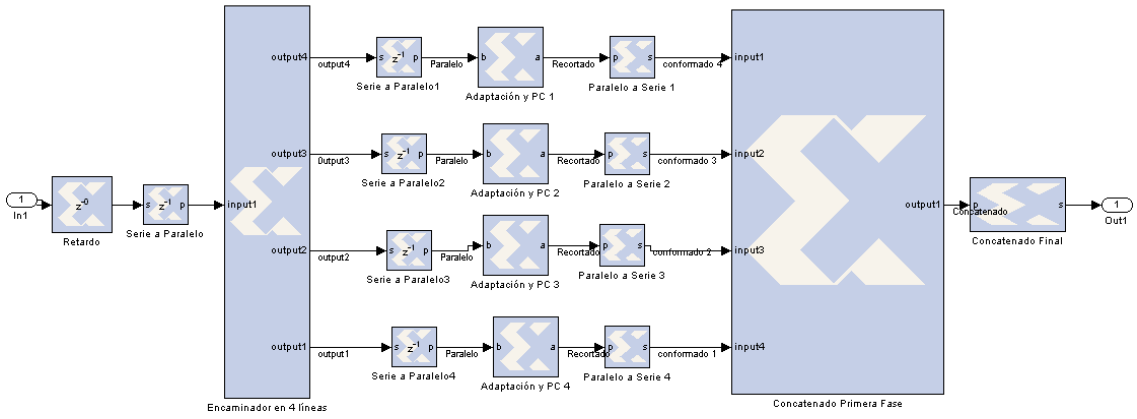

Figura 134. Modelo diseñado para Inserción de Prefijo Cíclico

El diseño se ha elaborado para un prefijo cíclico que verifique G = Tg / Tb = 1/8. En este caso, 32 muestras sobre las 256 del símbolo.

La operación ha de realizarse sobre el símbolo completo, lo que obliga a operar sobre él en paralelo. Al igual que en otros elementos explicados anteriormente, el número de bits que un bloque *Xilinx ® Serial to Parallel* es limitado, lo que puede obligar a emplear el "árbol de paralelización" para codificaciones mayores que BPSK.

El bloque descrito realiza de forma adicional el proceso de conformado de la salida de la IFFT, tal y como se describe en el apartado 3.5.16, en el que se describe el bloque *Xilinx ® FFT v1.0*

El volumen de datos que se maneja a la entrada (3 veces el tamaño de un símbolo OFDM), obliga en este caso a emplear el "árbol de paralelización" al superarse el número de bits que el bloque *Xilinx ® Serial to Parallel* puede manejar.

El tamaño de la trama a manejar es de 3x256 símbolos BPSK (a tasa triple) de 10 bits cada uno (7680 bits). De ellos, solo los 256 últimos símbolos BPSK son válidos.

Cada línea del árbol maneja 1920 bits, de los que se extraen los últimos 640 bits (256/4 = 64 símbolos BPSK de 10 bits) y concatena delante de estos los últimos (32/4 = 8 símbolos BPSK de 10 bits).

Las figuras 135 y 136 muestran la configuración de los principales bloques.

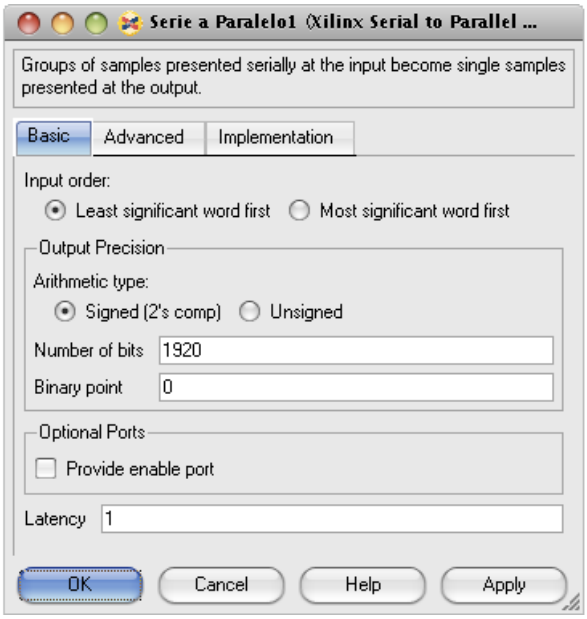

Figura 135. Configuración del Bloque Serie a Paralelo del "árbol de paralelización" del bloque de inserción de prefijo cíclico

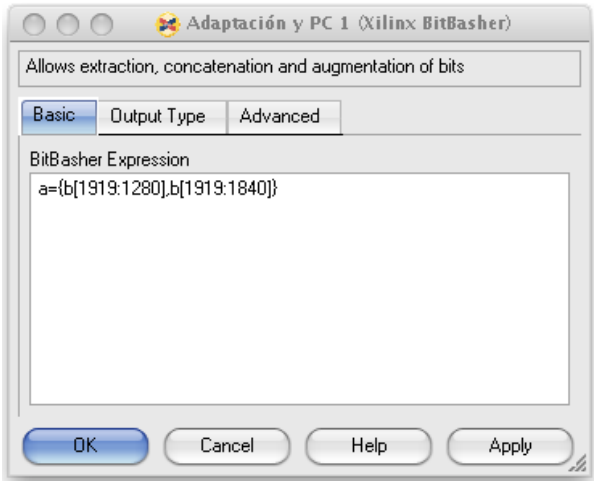

Figura 136. Configuración del bloque Bitbasher "Acondicionamiento y PC" del bloque de inserción de prefijo cíclico

5. VALIDACIÓN, SIMULACIÓN Y RESULTADOS

A continuación se muestra una captura de pantalla de la información recabada mediante un bloque *Xilinx WaveScope* en el que se muestra la evolución de las señales a través del emisor en todos los puntos relevantes.

La validación del emisor diseñado en este proyecto ha sido realizada mediante dos fuentes de datos introducidas en el emisor y la observación de las correspondientes salidas en el receptor.

a) Fuente de 1 bit: contador libre ascendente de 1 bit (0, 1, 0, 1…).

b) Fuente de 8 bits: contador libre ascendente de 8 bits (0, 1, 2,…, 254, 255, 0, 1, 2…).

Ambas fuentes de datos son las que proporcionan la información a la **trama** empleada, que se **repite cíclicamente** y consta de **10 símbolos OFDM**. La salida del emisor será el resultado de ejecutar todas las funciones descritas en el apartado anterior.

Para comprobar la validez del diseño realizado, se analiza y verifica en diferentes puntos del receptor (que se indican en paréntesis en cada apartado), para ambas fuentes, la correspondencia de la señal con su equivalente en el emisor. Los puntos de comparación seleccionados se muestran en la figura 137 (emisor) y 138 (receptor).

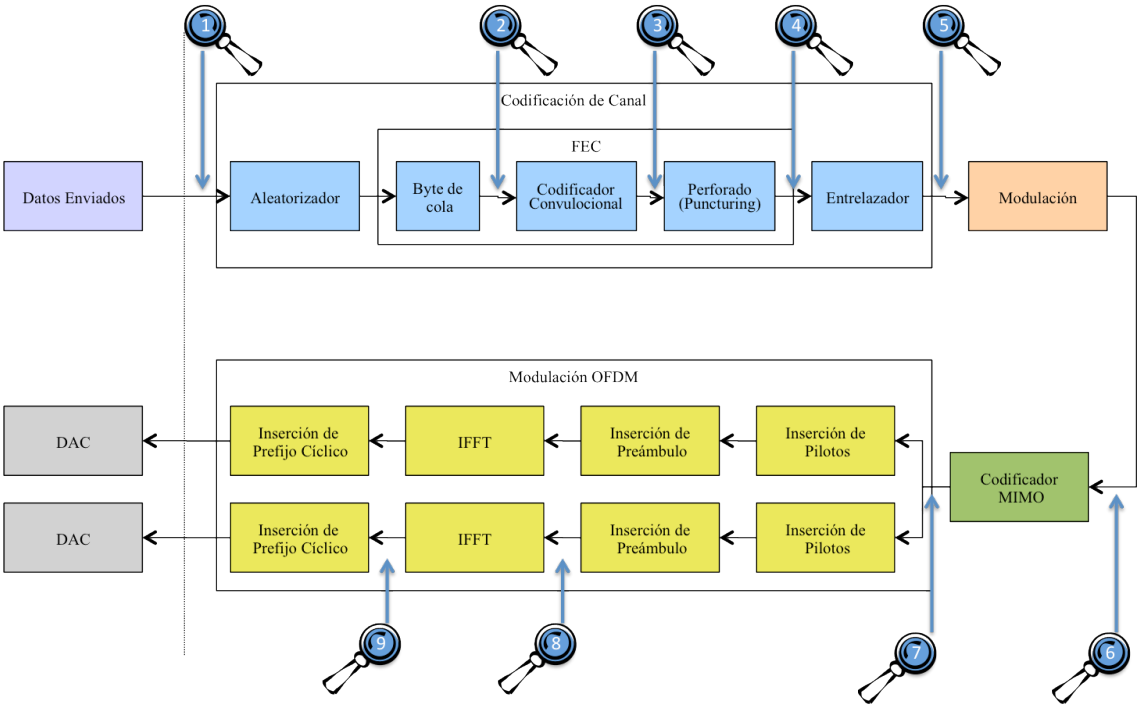

Figura 137. Puntos de Observación en el emisor.

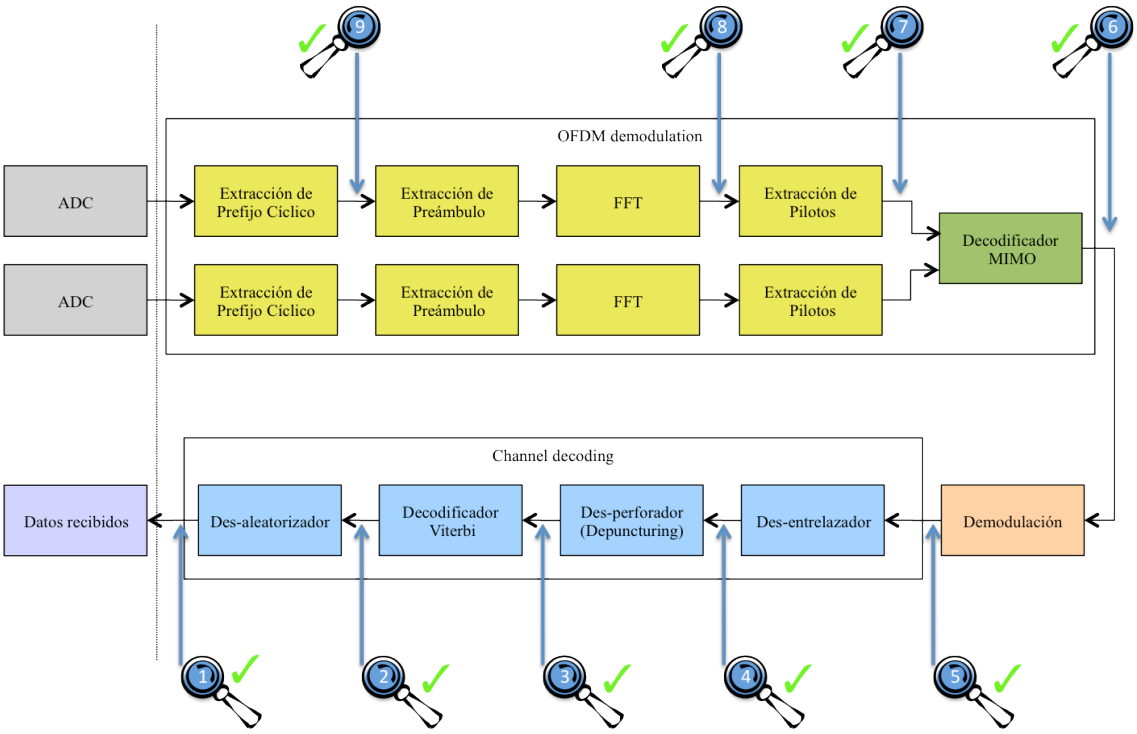

Figura 138. Puntos de Control en el receptor.

En los siguientes apartados, se muestran las capturas de pantalla de los resultados obtenidos empleando un bloque Xilinx WaveScope. Dichos resultados se corresponden con las señales adecuadas. Para facilitar su comprensión, todas las señales se muestran al comienzo de la trama y en formato hexadecimal.

Debe indicarse que el sistema funciona continuamente por lo que muchos bloques funcionan con datos no válidos (señal a 0) hasta que les llega la señal valida, que ha sufrido retardos por efecto de los bloques precedentes.

Por ello, se emplea una señal de control de errores que indica mediante su puesta a 0 que el bloque bajo comprobación comienza a trabajar con datos válidos.

5.1 Comprobación “Extremo a Extremo”

La última operación que se realiza en el receptor es la desaletorización. Tras ella, la señal obtenida deberá corresponderse con la introducida al emisor. Por tanto, comparación de la señal en ese punto deberá indicar el buen comportamiento del sistema completo.

Debe señalarse previamente que los bits que conforman la señal tras la acción del Desaleatorizador deben ser agrupados en un número adecuado (según el tipo de fuente) para recuperar los datos introducidos en el emisor. En este caso, puesto que una de las fuentes es

de 1 bit, sólo será necesario agruparlos para la fuente de 8 bits mediante un bloque *Xilinx Serial to Parallel*.

De esta manera la señal “Errores Desaleatorizador” indicará el momento a partir del cual los datos obtenidos a la salida son correctos, estableciéndose al valor 0.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las cuales se observa la correspondencia entre las señales del emisor y receptor (figuras 139, 140, 141,142, 143 y 144):

a) Fuente de 1 bit

Figura 139. Validación extremo a extremo. Forma general de la señales de entrada y salida con fuente de 1 bit

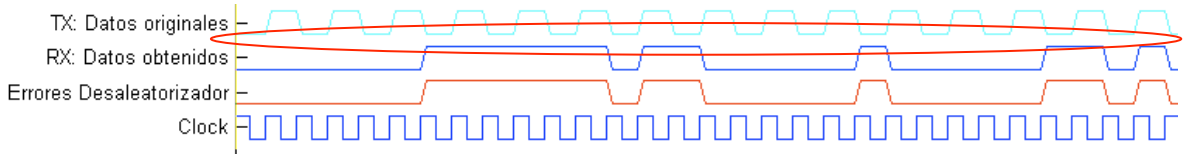

Figura 140. Validación extremo a extremo. Muestra de los datos en el emisor con fuente de 1 bit

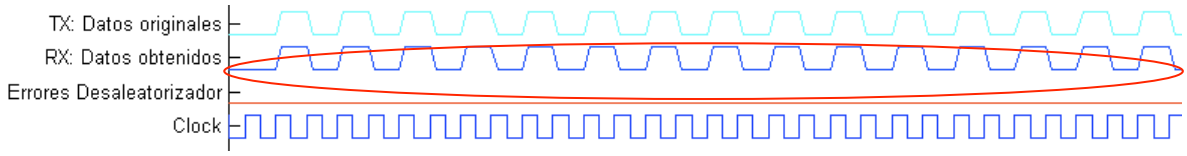

Figura 141. Validación extremo a extremo. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

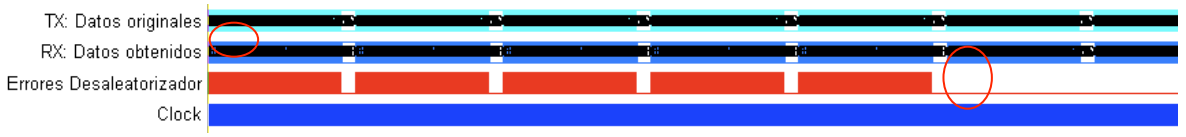

Figura 142. Validación extremo a extremo Forma general de la señales de entrada y salida con fuente de 8 bits

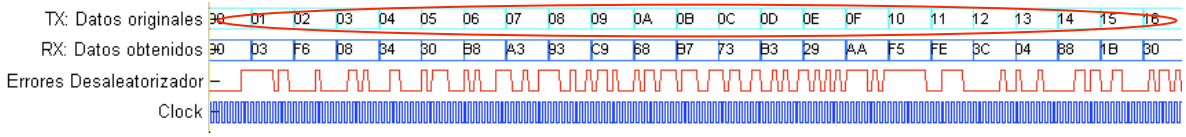

Figura 143. Validación extremo a extremo Muestra de los datos en el emisor con fuente de 8 bits

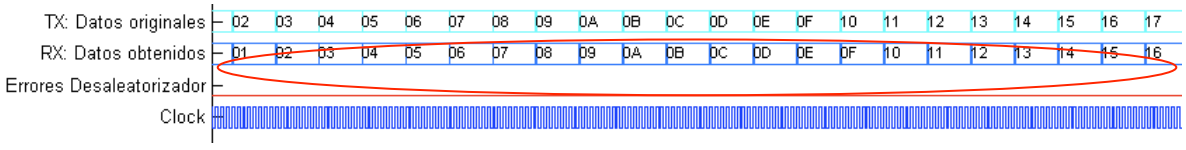

Figura 144. Validación extremo a extremo. Muestra de los datos en el receptor con fuente de 8 bits

5.2 Aleatorizador (en el desaleatorizador)

La señal a la salida del desaleatorizador esta constituida por una única línea, dado que las operaciones se realizan a nivel de bit.

Las figuras 145, 146, 147, 148, 149, 150 muestran las capturas realizadas al principio de la trama para ambas fuentes empleadas en todo el proceso de validación. En ellas pueden observarse la correspondencia entre las señales a la entrada del aleatorizador del emisor y la salida del desaleatorizador del receptor.

Debe señalarse que la línea “Errores Desaleatorizador” indica el comienzo del flujo de datos válidos mediante la puesta del valor a 0.

a) Fuente de 1 bit

Figura 145. Validación del Aleatorizador. Forma general de la señales de entrada y salida con fuente de 1 bit

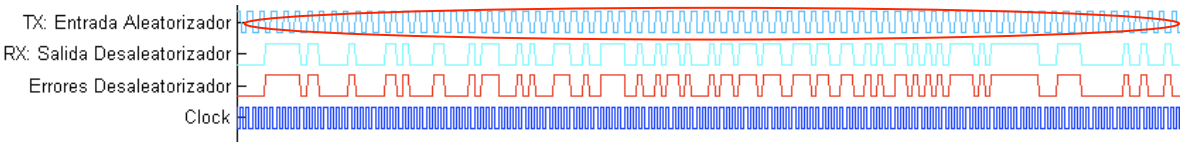

Figura 146. Validación del Aleatorizador. Muestra de los datos en el emisor con fuente de 1 bit

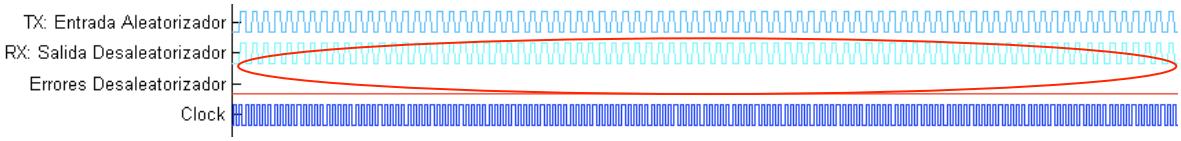

Figura 147. Validación del Aleatorizador. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 148. Validación del Aleatorizador. Forma general de la señales de entrada y salida con fuente de 8 bits

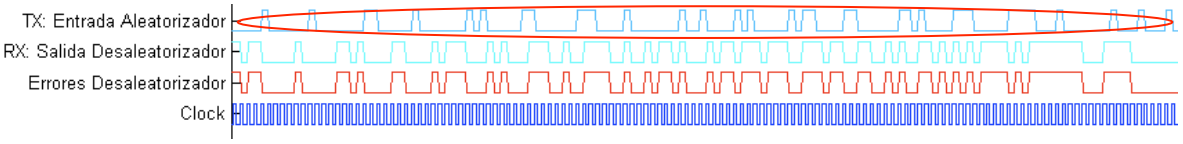

Figura 149. Validación del Aleatorizador. Muestra de los datos en el emisor con fuente de 8 bits

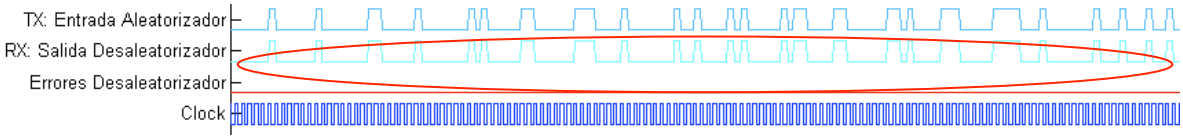

Figura 150. Validación del Aleatorizador. Muestra de los datos en el receptor con fuente de 8 bits

5.3 Codificador Convolucional (en el Decodificador Viterbi)

El decodificador viterbi invierte las operaciones efectuadas por el codificador convolucional La señal, tras la acción del Decodificador Viterbi se encuentra formada por única línea.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes. En ellas se puede observar la correspondencia entre las señales del emisor y receptor (figuras 151, 152, 153, 154, 155 y 156):

a) Fuente de 1 bit

Figura 151. Validación del Codificador Convolucional. Forma general de la señales de entrada y salida con fuente de 1 bit

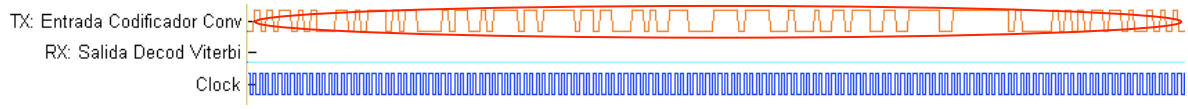

Figura 152. Validación del Codificador Convolucional. Muestra de los datos en el emisor con fuente de 1 bit

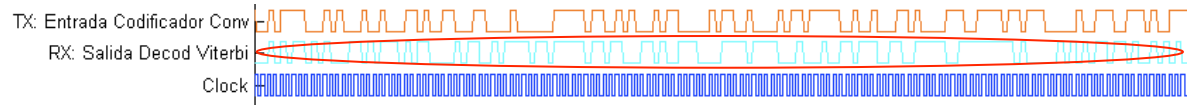

Figura 153. Validación del Codificador Convolucional. Muestra de los datos en el receptor con fuente de 1 bit

b) Fuente de 8 bits

Figura 154. Validación del Codificador Convolucional Forma general de la señales de entrada y salida con fuente de 8 bits

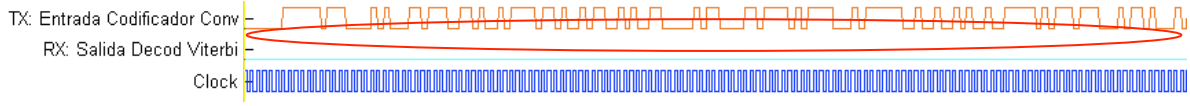

Figura 155. Validación del Codificador Convolucional. Muestra de los datos en el emisor con fuente de 8 bits

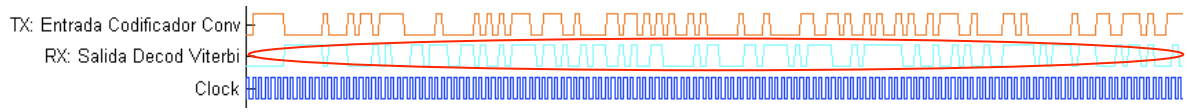

Figura 156. Validación del Codificador Convolucional Muestra de los datos en el receptor con fuente de 8 bits

5.4 Perforado – Puncturing (en el Des-perforador – Depuncturing)

El bloque de desperforado (depuncturing) invierte las operaciones efectuadas por el bloque de perforado (puncturing) La señal tras la acción del bloque de desporfarado se encuentra formada por 2 líneas (componente X e Y), preparadas para ser introducidas en el Decodificador Viterbi.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las que se puede observar la correspondencia entre las señales del emisor y receptor (figuras 157, 158, 159, 160, 161 y 162):

a) Fuente de 1 bit

Figura 157. *Validación del bloque de perforado (Depuncturing). Forma general de la señales de entrada y salida con fuente de 1 bit*

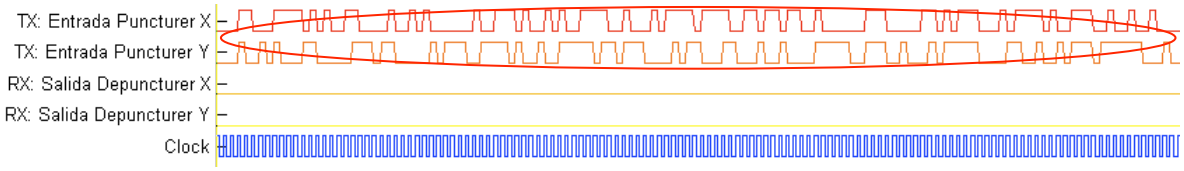

Figura 158. *: Validación del bloque de perforado (Depuncturing).). Muestra de los datos en el emisor con fuente de 1 bit*

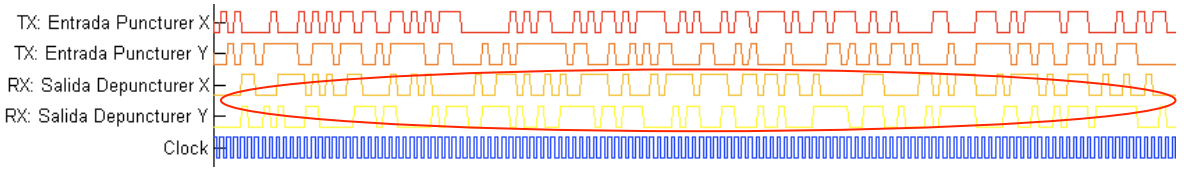

Figura 159. *Validación del bloque de perforado (Depuncturing). Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

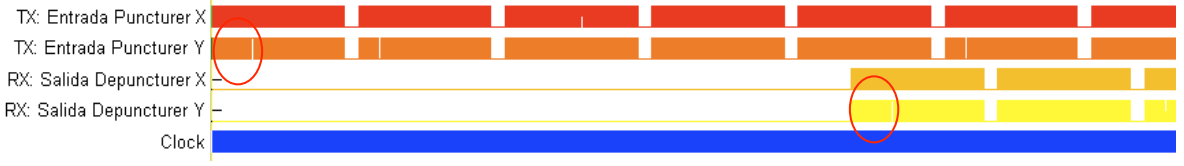

Figura 160. *Validación del bloque de perforado (Depuncturing). Forma general de la señales de entrada y salida con fuente de 8 bits*

-

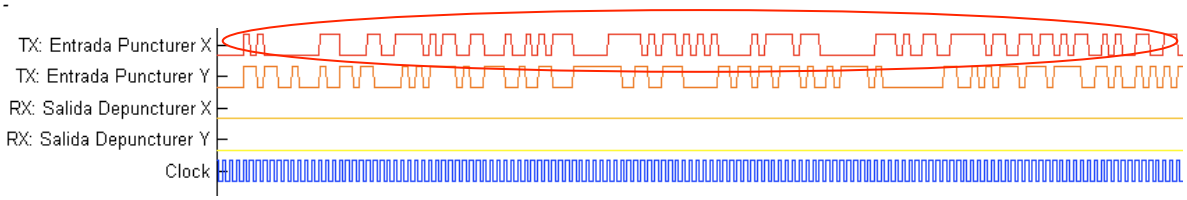

Figura 161. *Validación del bloque de perforado (Depuncturing). Muestra de los datos en el emisor con fuente de 8 bits*

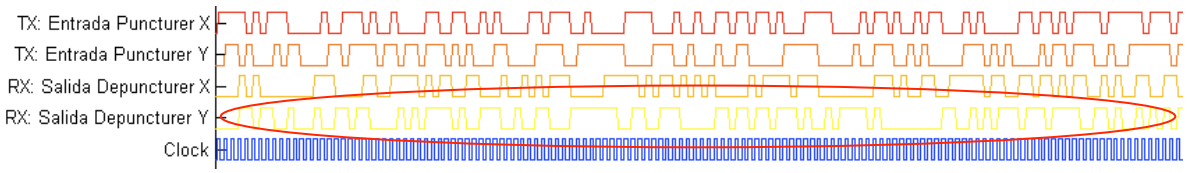

Figura 162. *Validación del bloque de perforado (Depuncturing). Muestra de los datos en el receptor con fuente de 8 bits*

5.5 Entrelazador (en el Desentralazador)

El Desentralezador invierte las operaciones efectuadas por el entralazador. Dado que estos bloques solo efectúan operaciones de mezclado de bits, solo se maneja una línea a la entrada y salida.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las que se puede observar la correspondencia entre las señales del emisor y receptor (figuras 163, 164, 165, 166, 167 y 168):

a) Fuente de 1 bit

Figura 163. *Validación del Entrelazador. Forma general de la señales de entrada y salida con fuente de 1 bit*

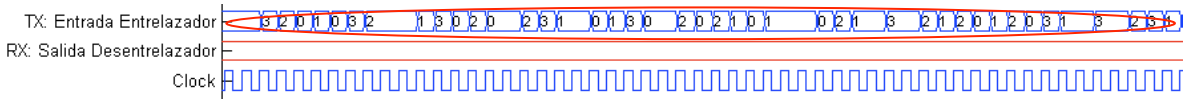

Figura 164. *Validación del Entrelazador. Muestra de los datos en el emisor con fuente de 1 bit*

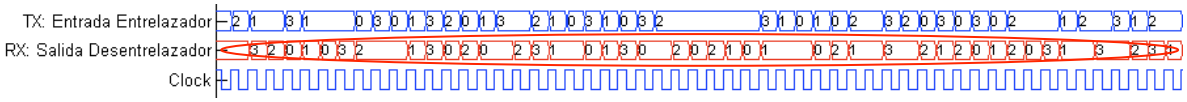

Figura 165. *validación del Entrelazador. Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

Figura 166. *Validación del Entrelazador. Forma general de la señales de entrada y salida con fuente de 8 bits*

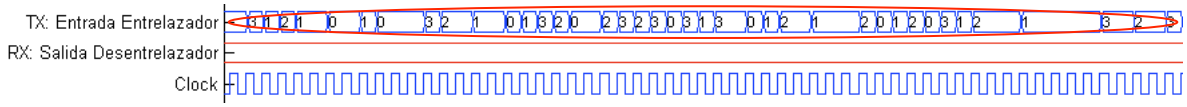

Figura 167. *Validación del Entrelazador. Muestra de los datos en el emisor con fuente de 8 bits*

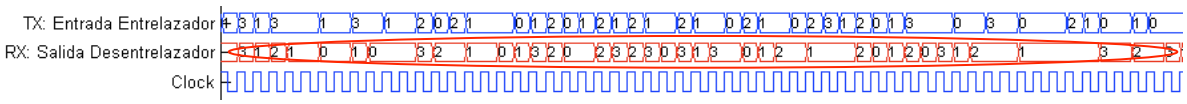

Figura 168. *Validación del Entrelazador. Muestra de los datos en el receptor con fuente de 8 bits*

5.6 Modulador (en el Demodulador)

El Demodulador invierte las operaciones efectuadas por el Modulador, reuniendo las dos componentes I y Q que genera este último en una única línea que alimenta al desentralazador. Desde el punto de vista del emisor, todas las operaciones que realizan los bloques siguientes se realizan sobre las dos componentes de la señal (I y Q). Debe tenerse en cuenta que el emisor desarrollado emplea modulación BPSK, por lo que la componente Q de la señal, deberá permanecer a 0.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las que se puede observar la correspondencia entre las señales del emisor y receptor (figuras 169, 170, 171, 172, 173 y 174):

a) Fuente de 1 bit

Figura 169. *Validación del Modulador. Forma general de la señales de entrada y salida con fuente de 1 bit*

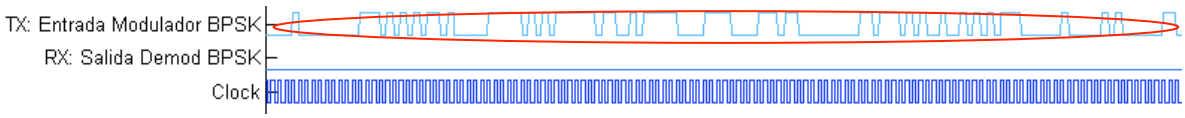

Figura 170. *Validación del Modulador. Muestra de los datos en el emisor con fuente de 1 bit*

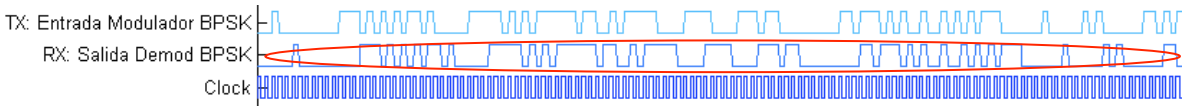

Figura 171. *Validación del Modulador. Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

Figura 172. *Validación del Modulador. Forma general de la señales de entrada y salida con fuente de 8 bits*

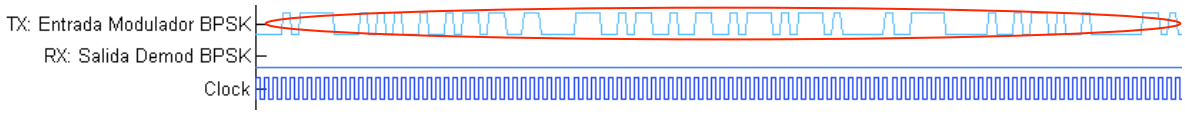

Figura 173. *Validación del Modulador. Muestra de los datos en el emisor con fuente de 8 bits*

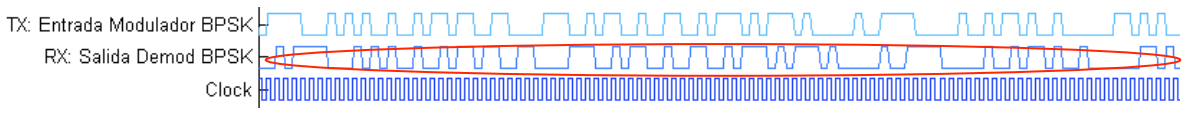

Figura 174. *Validación del Modulador. Muestra de los datos en el receptor con fuente de 8 bits*

5.7 Codificador MIMO (en el Decodificador MIMO)

El Decodificador MIMO invierte las operaciones efectuadas por el Codificador MIMO. El Decodificador MIMO reúne las dos líneas independientes (con componentes I y Q cada una) que genera el Codificador MIMO para alimentar sendas antenas del emisor. Desde el punto de vista del emisor, todas las operaciones de los siguientes bloques se realizan sobre cada una de estas dos líneas independientes por separado.

Las siguiente figuras (figuras 175, 176, 177, 178, 179 y 180) muestran las capturas realizadas al principio de la trama para ambas fuentes, a fin de observar la correspondencia entre las señales del emisor y receptor.

a) Fuente de 1 bit

Figura 175. *Validación del Codificador MIMO. Forma general de la señales de entrada y salida con fuente de 1 bit*

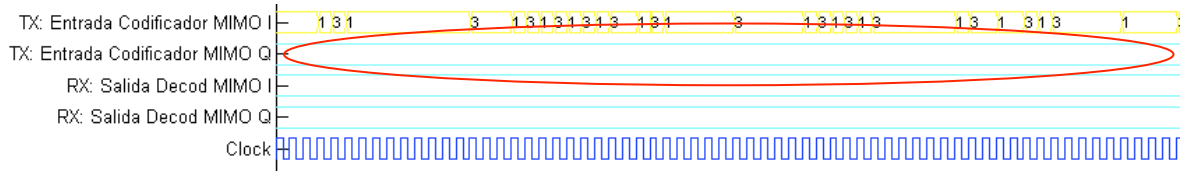

Figura 176. *Validación del Codificador MIMO. Muestra de los datos en el emisor con fuente de 1 bit*

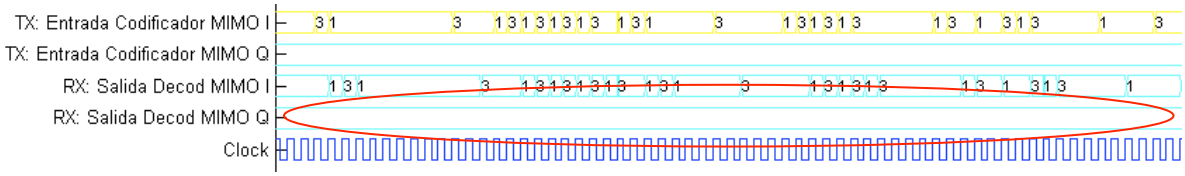

Figura 177. *Validación del Codificador MIMO. Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

Figura 178. *Validación del Codificador MIMO. Forma general de la señales de entrada y salida con fuente de 8 bits*

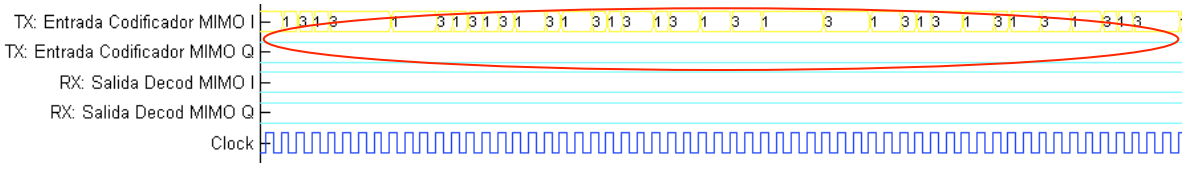

Figura 179. *Validación del Codificador MIMO. Muestra de los datos en el emisor con fuente de 8 bits*

Figura 180. *Validación del Codificador MIMO. Muestra de los datos en el receptor con fuente de 8 bits*

5.8 Inserción de pilotos (en el bloque de Extracción de pilotos)

El bloque de Extracción de pilotos invierte las operaciones efectuadas por el bloque de Inserción de Pilotos. Estas operaciones se realizan para las componentes I y Q de cada una de las dos líneas independientes. Se manejan, por tanto, cuatro líneas de datos tanto a la entrada como a la salida de dichos bloques.

Las figuras 181, 182, 183, 184, 185 y 186 muestran la correspondencia entre las señales del emisor y receptor, mediante las capturas realizadas al comienzo de la trama para ambas fuentes.

a) Fuente de 1 bit

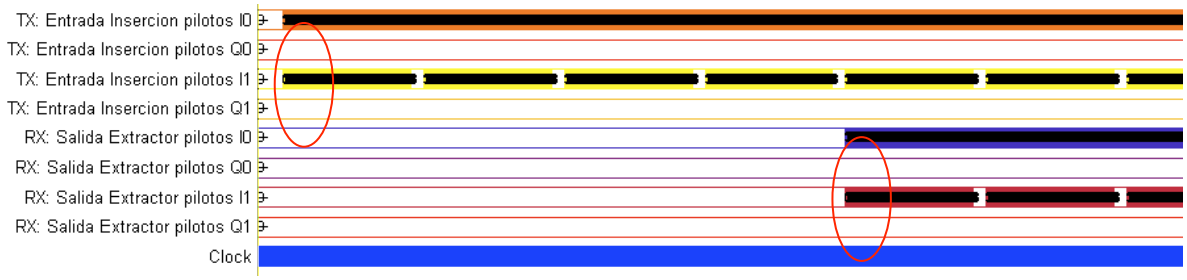

Figura 181. *Validación del bloque de Insercción de pilotos. Forma general de la señales de entrada y salida con fuente de 1 bit*

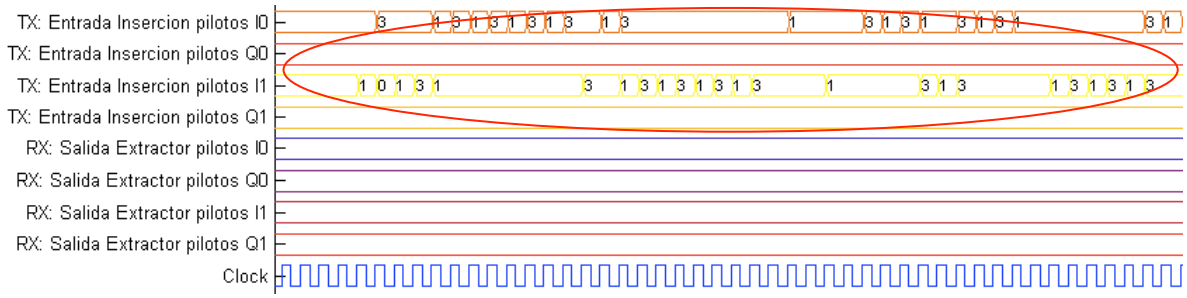

Figura 182. *Validación bloque de Insercción de pilotos. Muestra de los datos en el emisor con fuente de 1 bit*

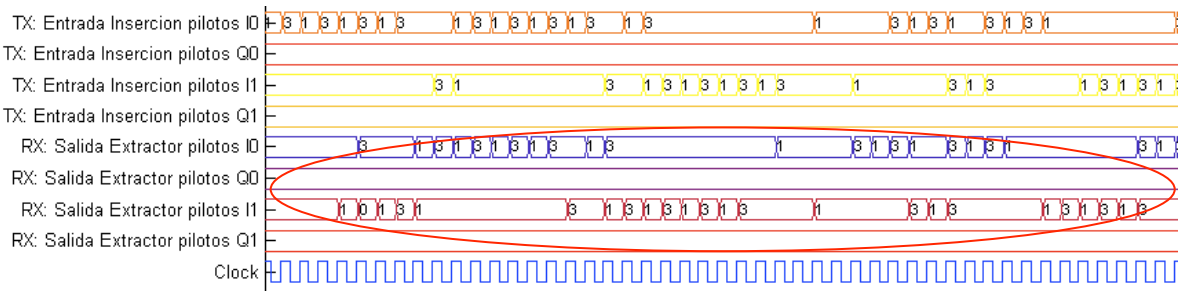

Figura 183. *Validación del bloque de Insercción de pilotos. Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

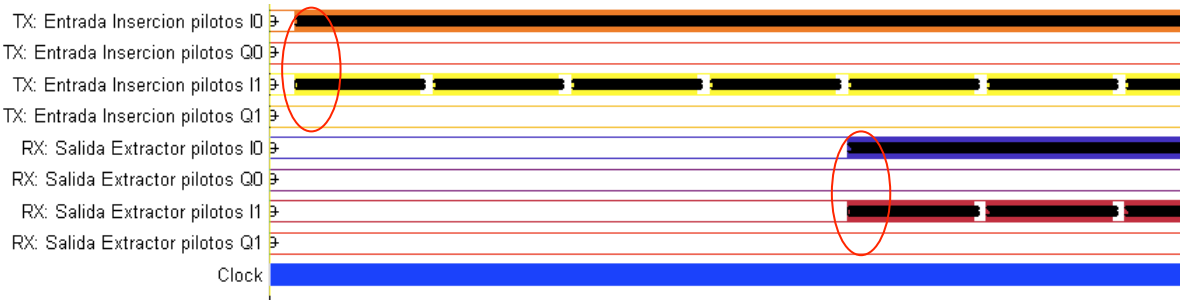

Figura 184. *Validación del bloque de Insercción de pilotos. Forma general de la señales de entrada y salida con fuente de 8 bits*

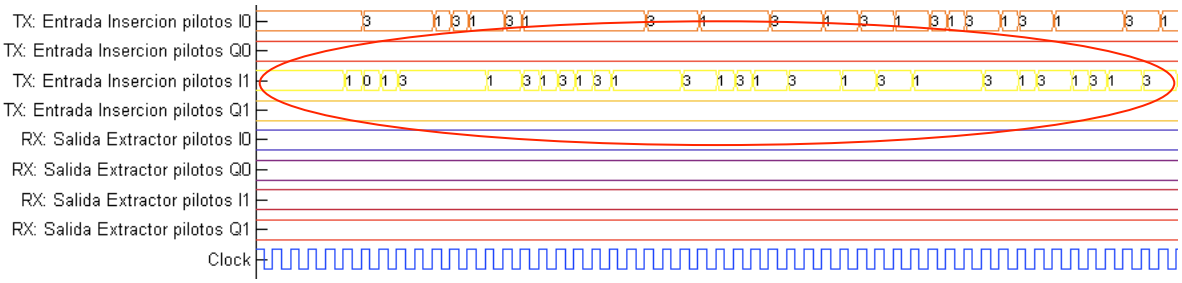

Figura 185. *Validación del bloque de Insercción de pilotos. Muestra de los datos en el emisor con fuente de 8 bits*

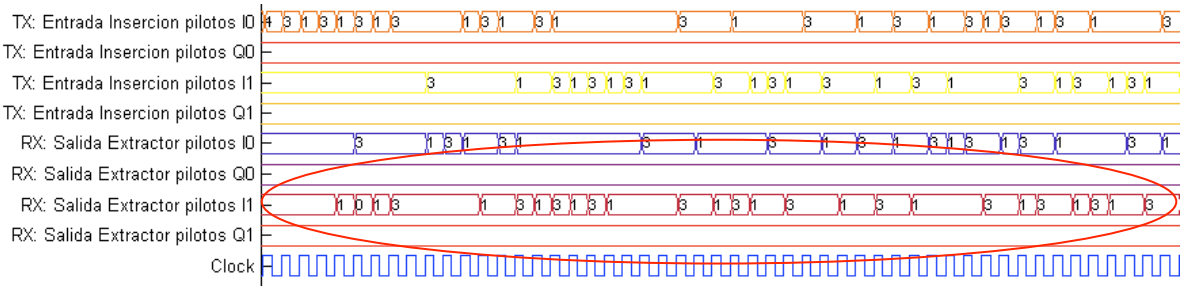

Figura 186. *Validación del bloque de Insercción de pilotos. Muestra de los datos en el receptor con fuente de 8 bits*

5.9 Inserción de preámbulo e IFFT (tras el bloque FFT)

Se considera conveniente validar el bloque de Inserción de Preámbulo junto con el B-loque IFFT, debido a que en el bloque receptor las operaciones se realizan en orden inverso al habitual.

Dado que en el emisor se inserta el preámbulo y a continuación se realiza la operación IFFT (con lo que dicha operación se ejecuta sobre la trama con el preámbulo), el orden lógico para deshacer dichas operaciones en el receptor sería ejecutar el bloque FFT y a continuación extraer el preámbulo.

Sin embargo, dado que el preámbulo tiene una forma conocida, no resulta necesario realizar la operación FFT sobre él. El preámbulo puede ser empleado para realizar las operaciones de alineación de la trama directamente.

Por tanto, no existe una correspondencia bloque a bloque entre las entradas en el emisor y sus correspondientes salidas en el receptor para cada uno de estos bloques por separado, sino que esta existe para la operación conjunta de ambos bloques.

Las operaciones de ambos bloques, se ejecutan sobre 4 líneas (componentes I y Q para ambas antenas. Dada la conversión espacio-tiempo que realiza la IFFT, la componente Q de cada una de las líneas deja de ser 0.

A continuación se muestran las capturas realizadas al principio de la trama para ambas fuentes, en las que se observa la correspondencia entre las señales del emisor y receptor (figuras 187, 188, 189, 190, 191 y 192).

Debe indicarse que, dado que para ambas fuentes el principio de la trama está compuesto por el preámbulo, la forma de la señal será idéntica para ambos casos.

a) Fuente de 1 bit

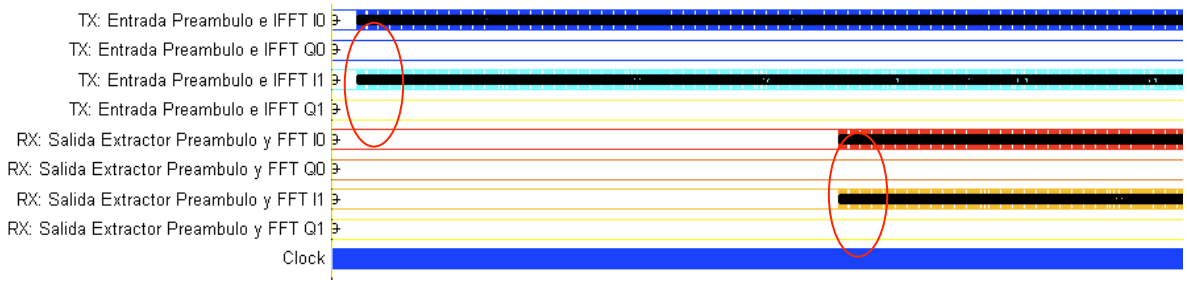

Figura 187. . *Validación de la Inserción del preámbulo e IFFT Forma general de la señales de entrada y salida con fuente de 1 bit*

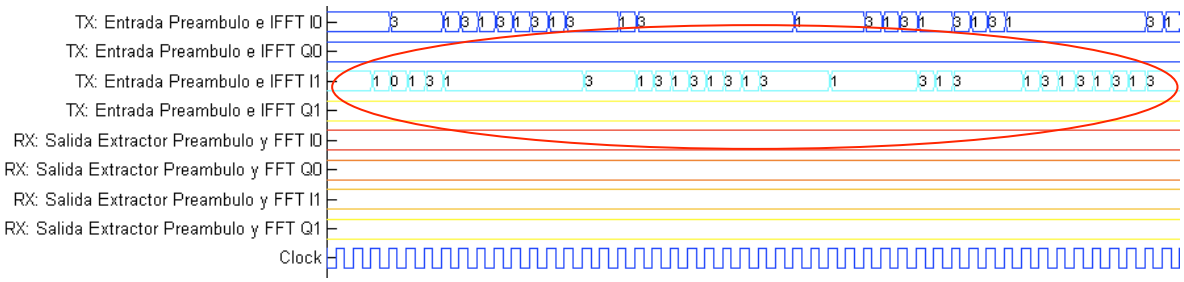

Figura 188. *Validación de la Inserción del preámbulo e IFFT. Muestra de los datos en el emisor con fuente de 1 bit*

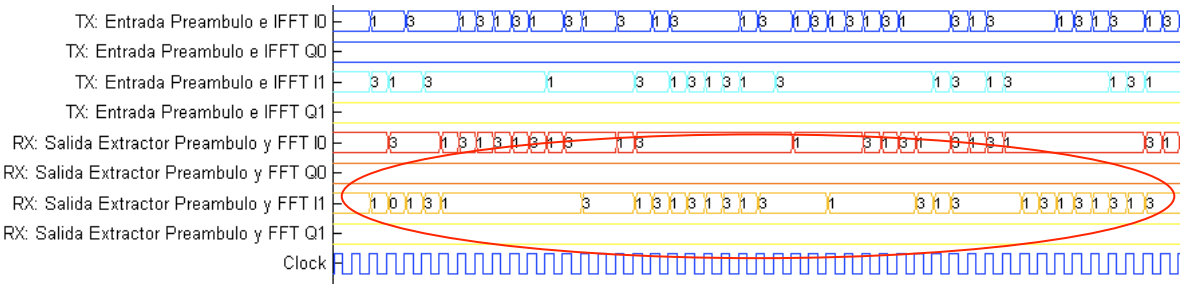

Figura 189. *Validación de la Inserción del preámbulo e IFFT. Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

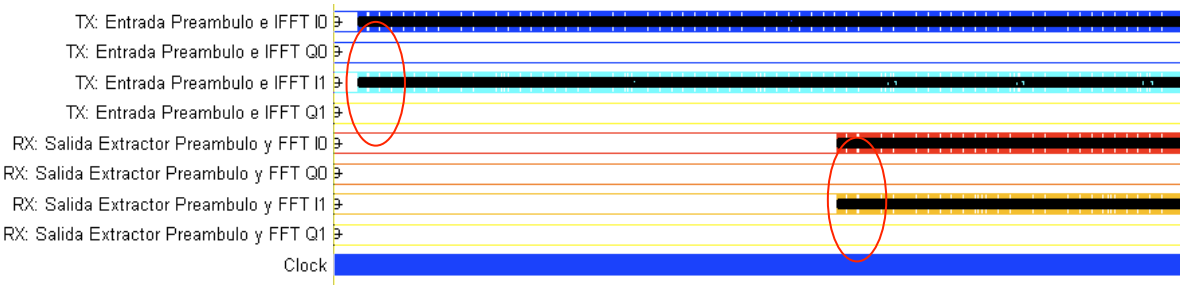

Figura 190. *Validación de la Inserción del preámbulo e IFFT. Forma general de la señales de entrada y salida con fuente de 8 bits*

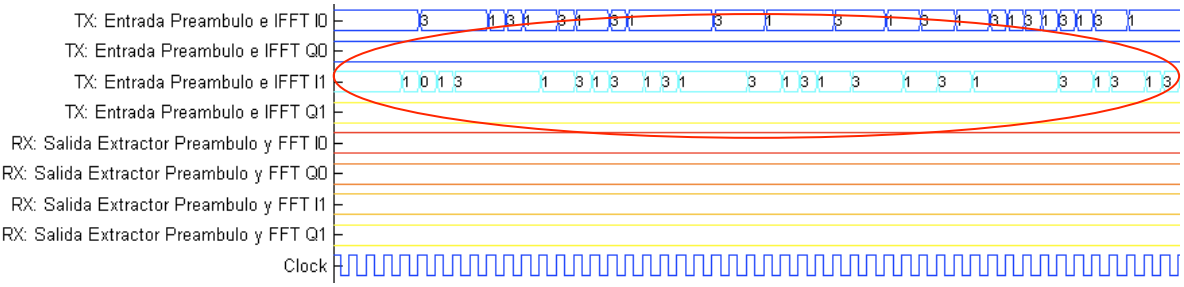

Figura 191. *Validación de la Inserción del preámbulo e IFFT. Muestra de los datos en el emisor con fuente de 8 bits*

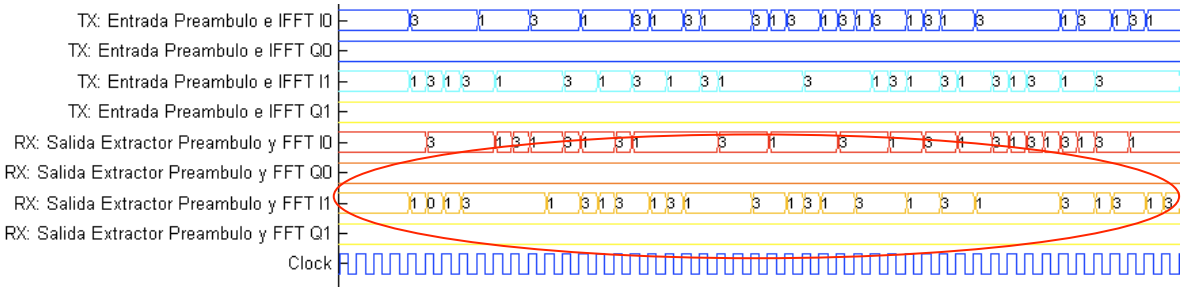

Figura 192. *Validación de la Inserción del preámbulo e IFFT. Muestra de los datos en el receptor con fuente de 8 bits*

5.10 Inserción de prefijo cíclico (en el bloque de Extracción de prefijo cíclico)

El bloque de Extracción de prefijo cíclico invierte las operaciones efectuadas por el bloque de Inserción de prefijo cíclico. Estas operaciones se realizan para las componentes I y Q de cada una de las dos líneas independientes. Se manejan, por tanto, cuatro líneas de datos tanto a la entrada como a la salida de dichos bloques.

Las figuras 193, 194, 195, 196, 197 y 198 muestran la correspondencia entre las señales del emisor y receptor, mediante las capturas realizadas al comienzo de la trama para ambas fuentes.

a) Fuente de 1 bit

Figura 193. *Validación del bloque de Inserción del prefijo cíclico. Forma general de la señales de entrada y salida con fuente de 1 bit*

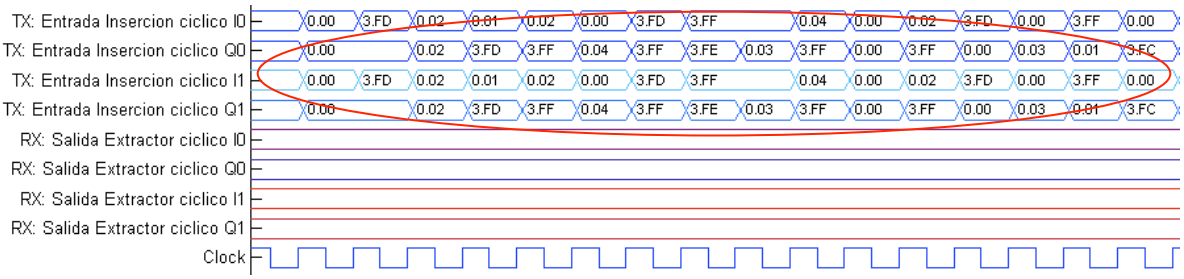

Figura 194. *Validación del bloque de Inserción del prefijo cíclico. Muestra de los datos en el emisor con fuente de 1 bit*

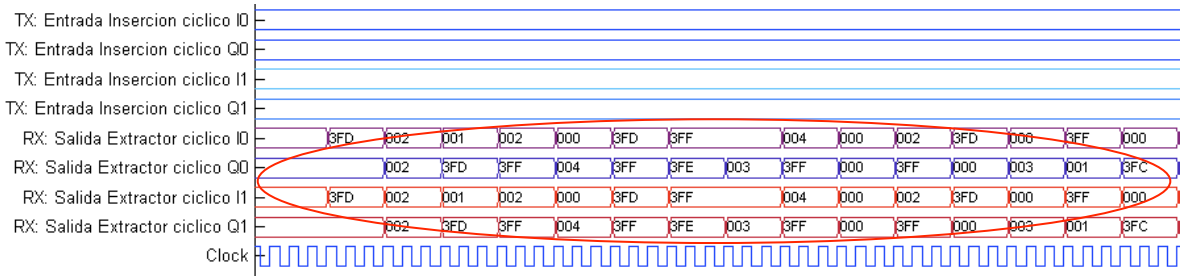

Figura 195. *Validación del bloque de Inserción del prefijo cíclico. Muestra de los datos en el receptor con fuente de 1 bit*

b) Fuente de 8 bits

Figura 196. *Validación del bloque de Inserción del prefijo cíclico. Forma general de la señales de entrada y salida con fuente de 8 bits*

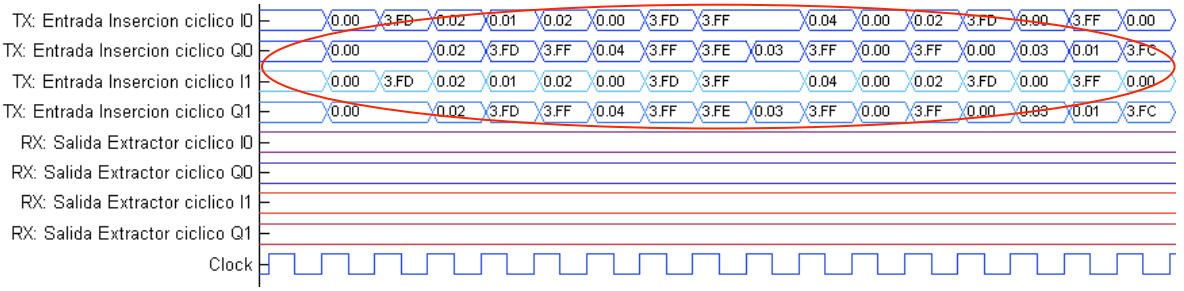

Figura 197. *Validación del bloque de Inserción del prefijo cíclico. Muestra de los datos en el emisor con fuente de 8 bits*

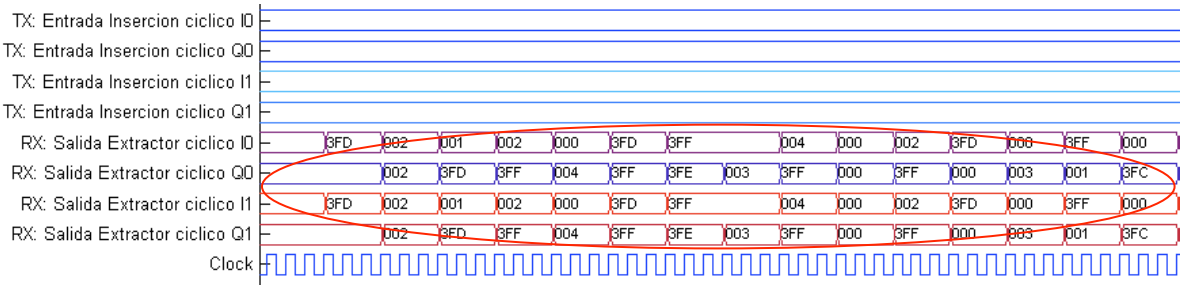

Figura 198. *Validación del bloque de Inserción del prefijo cíclico. Muestra de los datos en el receptor con fuente de 8 bits*

6. CONCLUSIONES Y FUTUROS TRABAJOS

La realización de este proyecto permite **extraer una serie de conclusiones, tanto positivas como referidas a aspectos mejorables detectados**. Adicionalmente, se han identificado futuras líneas de trabajo potenciales en relación con diferentes características de dicho proyecto.

Los siguientes apartados ilustran de manera detallada estas conclusiones y futuras líneas de trabajo.

6.1 Conclusiones

El cumplimiento de los objetivos que se fijaron al comienzo del proyecto ha implicado un proceso de aprendizaje y familiarización con diversas herramientas. De este trabajo, se pueden extraer una serie de conclusiones en relación con el diseño este emisor para el estándar IEEE 802.16e-2005 utilizando MIMO 2x2, OFDM en la capa física y BPSK para la modulación de datos. Estas conclusiones, tanto de carácter positivo como sobre aspectos mejorables, pueden servir como base para la realización de diferentes trabajos en el futuro.

Así, entre las principales conclusiones positivas destacan las siguientes:

- A la vista del estudio y análisis de la definición, características y aplicaciones el estándar IEEE 802.16e-2005 cuenta con importantes posibilidades de convertirse en una referencia en las comunicaciones inalámbricas de alta capacidad. Este hecho está en línea con la situación actual observada y prevista del panorama de las comunicaciones en el ámbito de los estándares inalámbricos.
- La combinación de las técnicas MIMO junto a la modulación OFDM dotan a los sistemas de comunicación de una robustez en escenarios multitrayecto y sin línea de visión directa que permite incrementar el radio de la celda y su capacidad. Esto la convierte en el futuro de los sistemas de comunicaciones inalámbricos de alta capacidad.
- El alto nivel de abstracción, la variedad de recursos, la posibilidad de simulación previa y la compilación automática del modelo FPGA (generando el código correspondiente), dotan a las plataformas de desarrollo para FPGA basadas en Simulink® de un gran potencial. Además, en el caso particular de este proyecto, se ha verificado que es posible la consecución de un prototipo sin una excesiva utilización de los diferentes recursos de la *VHS-ADC Virtex-4.*

En relación con los aspectos mejorables, principalmente centrados en la plataforma de desarrollo para FPGA sobre Simulink®, destacan los siguientes:

- La curva de aprendizaje de la plataforma es elevada y el proceso costoso y laborioso. La utilización y respuesta de los bloques no es todo lo intuitiva que se esperaría. Se hace necesario un estudio experimental de cada bloque previamente a su empleo en un sistema para asegurar una correcta interacción con el resto de bloques. Una vez se ha alcanzado un grado de familiaridad y conocimiento suficiente, el tiempo de desarrollo se reduce considerablemente respecto a otras plataformas no basadas en Simulink®.
- La variedad de bloques y flexibilidad de la plataforma resulta en ocasiones insuficiente para implementar operaciones o lógicas de carácter avanzado de una manera sencilla. Para solucionar esta carencia, este tipo de plataformas de desarrollo incrementan y mejoran de manera constante su catálogo de bloques compatibles con Simulink®.

6.2 Trabajos futuros

A partir de las conclusiones se pueden proponer las siguientes dos líneas de trabajo.

- Mejora de las prestaciones del emisor diseñado para el estándar IEEE 802.16d-2004 (WiMAX Fijo).
- Aplicación del conocimiento adquirido sobre la plataforma de desarrollo para FPGA basada en Simulink® (*VHS-ADC Virtex-4* de Lyrtech [2] junto a la herramienta *Xilinx System Generator for DSP* [3]) para el diseño de sistemas de comunicaciones basados en otros estándares.

Así mismo, entre las principales mejoras al modelo empleado se proponen una serie de mejoras:

- Diseñar toda la lógica necesaria para permitir emplear, además de BPSK, cualquiera de las constelaciones contempladas por el estándar IEEE 802.16e-2005 para la modulación de datos: QPSK, 16-QAM y 64-QAM. I
- Agregar la lógica asociada a la existencia de un canal de comunicaciones con el objetivo de estimarlo y compensar sus efectos.
- Mejorar las lógicas de retardo para optimizar el empleo de recursos en las FPGA. Este aspecto es especialmente importante en ciertos bloques que necesitan retardar grandes cantidades de datos. En estos bloques se debería estudiar la posibilidad de emplear alternativamente bloque de memoria RAM o FIFO.
- Además, el empleo de otras capas físicas incluidas en el estándar, como OFDMA, podría resultar especialmente interesante.

Sin embargo, la actividad futura más relevante sea quizás la síntesis, ejecución y validación de dicho emisor sobre una FPGA. Este aspecto es relevante dado que se ha podido comprobar que no todos los bloques que conforman este modelo, una vez sintetizado e introducido en una FPGA, se comportan de la misma manera que durante su simulación. Así, se hace necesario ejecutar el modelo y capturar directamente datos de la FPGA, analizándolos para identificar la causa de este comportamiento diferencial.

De esta forma, se concluiría el ciclo completo descrito en los objetivos del presente documento, cubriendo el punto 3 del mismo. En este sentido, se deben garantizar dos cosas:

- Que en la síntesis realizadas con Xilinx® System Generator e ISE de Xilinx® se cumplan las restricciones de la FPGA (área, timings, etc.)
- Que la simulación post place and route coincida con la simulación funcional.

Debe destacarse por último que el modelo que se ha empleado como prototipo para la elaboración de este proyecto contiene una versión del emisor y receptor, por lo que no está diseñado para el traslado directo a una FPGA, dado que los bloques del emisor se deberían trasladar al sistema equipado con conversor DAC y el receptor al equipo con conversor ADC. Esta implantación por separado tiene sentido una vez se conecte el sistema a las antenas y se implementen los bloques de estimación de canal.

Así mismo, debe tenerse en cuenta que el modelo empleado contiene toda la lógica necesaria para realizar las comprobaciones y comparaciones de señal oportunas, que aportan un grado de complejidad, principalmente en forma de retardos, que resulta innecesario en sistema real.

7. PRESUPUESTO

Este apartado mostrará el coste de los recursos, personales y materiales, necesarios para la realización del presente proyecto.

Para los siguientes cálculos, se considerará que la duración del proyecto ha sido equivalente a 10 meses a dedicación completa.

7.1 Coste material

La realización de este proyecto ha contado con los siguientes recursos materiales:

- Dos ordenadores portátiles personales valorados en 1.800 € y 800 € que incluyen el sistema operativo Windows XP. Estos ordenadores han sido empleados para otras funciones durante el tiempo del proyecto, antes y después del mismo. Asumiendo una vida útil para los ordenadores de 3 años (36 meses), y dado el

empleo mixto de los mismos, podemos estimar un coste aproximado de 350 € para el primero y 160 € para el segundo.

- Plataforma de desarrollo sobre FPGA basada en Simulink® denominada *VHS-ADC Virtex-4* de Lyrtech [2] valorada en 18.000 €. Dicha plataforma viene integrada en un ordenador con el sistema operativo Windows XP incluido. Debido a que el equipo será empleado en otros proyectos del departamento, estimamos el coste imputable al proyecto en unos 3.000 €.
- Herramienta software matemática MATLAB® (incluye Simulink®) [12] cuya licencia ronda los 3.000 €. Dicha licencia es compartida con el resto del departamento y seguirá siendo empleada. Por tanto, el coste imputable es de unos 125 € por equipo, es decir, 375 € en total.
- Herramienta software *Xilinx System Generator for DSP* [3] cuya licencia tiene un coste aproximado de 6.000 €. Dicha seguirá siendo empleada por el departamento. Por tanto, el coste imputable es de unos 250 € por equipo, es decir, 750 € en total.
- Conexión a Internet con un coste de unos 50 € / mes. Durante los 10 meses de duración del proyecto el proyecto ha empelado el 5% de la capacidad de la conexión. Por tanto, el coste imputable al proyecto es de unos 2,5 €/mes, 25 euros en total.
- Impresión de documentos con un coste aproximado de 100 €.

Así, el **coste material** de la realización del proyecto es el mostrado en la tabla 3:

Tabla 3. Coste material de la realización del proyecto

Concepto	Coste imputable (€)
Ordenadores portátiles con Windows XP	510
VHS-ADC Virtex-4 de Lyrtech con Windows XP	3.000
MATLAB® y Simulink®	375
Xilinx System Generator for DSP	750
Conexión Internet	25
Impresión documentos	100
TOTAL (Coste material)	**4.760**

7.2 Coste personal

La realización de este proyecto ha contado con los siguientes recursos personales:

- Desarrollador del proyecto con un coste laboral de 70 € / hora (según el Colegio de Ingenieros de Telecomunicación [14]). Por tanto, los 10 meses (21 días laborables por mes) de ejecución del proyecto con una dedicación media de 4 horas / día suponen un coste de 58.800 €.
- Directores del proyecto (Dr. Víctor P. Gil Jiménez y Dr. Enrique San Millán Heredia) con un coste laboral de 70 € / hora (según el Colegio de Ingenieros de Telecomunicación [14]) y una dedicación de unas 100 horas entre los dos. Por tanto, el coste imputable es de 7.000 €.

Así, el **coste personal** de la realización del proyecto es el mostrado en la tabla 4:

Tabla 4. *Coste personal de la realización del proyecto*

Concepto	Coste imputable (€)
Desarrollador proyecto	58.800
Directores proyecto	7.000
TOTAL (Coste personal)	**65.800**

7.3 Coste total

La realización de este proyecto tiene un **coste total** que debe incluir los costes materiales y personales, tal y como se muestra en la tabla 5:

Tabla 5. *Coste total de la realización del proyecto*

Concepto	Coste imputable (€)
Coste material	4.760
Coste personal	65.800
TOTAL	**70.560**

8. BIBLIOGRAFÍA

[1] "Air Interface for Fixed and Mobile Broadband Wireless Access Systems", IEEE STD 802.16e-2005, febrero 2006.

[2] "Air Interface for Fixed Broadband Wireless Access Systems", IEEE STD 802.16-2004, octubre 2004.

[3] Sitio Web de Xilinx. Último acceso en noviembre 2010.

[4] "Estudio práctico sobre el prototipado de un sistema MIMO 2x2 para WiMAX", David Díaz Martín y Roberto Prieto Alonso, Estudio Tecnológico de la Universidad Carlos III de Madrid, diciembre 2007.

[5] Sitio Web del WiMAX Forum, http://www.WiMAXforum.org/. Último acceso en en octubre 2010.

[6] Sitio Web de Lyrtech, http://www.lyrtech.com/. Último acceso en noviembre 2010.

[7] Sitio Web de Sociedad de la Información de la Fundación Telefónica, http://sociedadinformacion.fundacion.telefonica.com. Último acceso en noviembre 2010.

[8] Sitio Web del Observatorio Tecnológico del Ministerio de Educación, http://observatorio.cnice.mec.es/. Último acceso en octubre 2010.

[10] Sitio Web de la Universidad Galileo de Guatemala, http://www.galileo.edu/. Último acceso en octubre 2010.

[11] Sitio Web de Intel sobre WiMAX, http://www.intel.com/technology/WiMAX/. Último acceso en mayo de 2009. Último acceso en noviembre 2010.

[12] Sitio Web de The MathWorks, http://www.mathworks.com/. Último acceso en noviembre 2010.

[13] Herramienta web de despliegues WiMAX de WiMAX Forum http://www.WiMAXmaps.org. Último acceso en octubre 2010.

[14] Sitio Web del Colegio Oficial de Ingenieros de Telecomunicación, http://www.coit.es. Último acceso en enero de 2011.

[15] Sitio Web de la Fundación Telefónica, http://www.fundacion.telefonica.com. Último acceso en enero de 2011.

[15] Sitio Web de Iberbanda
http://www.iberbanda.es. Último acceso en junio de 2011.

9. ANEXO A: PARALELIZACIÓN

Algunos de los bloques presentes en el proyecto requieren realizar operaciones sobre los datos en forma de símbolo OFDM o trama completa. Dado el carácter síncrono del sistema y el carácter regular de la fuente de datos, estas operaciones pueden modificar la tasa binaria de información por las líneas de datos. En el caso del emisor, típicamente aumentan la tasa binaria al añadir datos a la línea (preámbulos, prefijos, etc.).

Pongamos como ejemplo que se quiere añadir un prefijo de cuatro bits a una trama de 16 bits, tal y como muestra la figura 199.

Figura 199. Anexo A. Árbol de Paralelización. Ejemplo. Operación a realizar.

Cualquier bloque que realice esta operación modificará la tasa binaria de la línea. Sea R_{be} la tasa binaria a la entrada de dicho bloque, la tasa binaria a la salida será $R_{bs} = R_{be} \bullet \frac{L_s}{L_e}$, siendo L_s y L_e las longitudes de la trama a la salida y la entrada respectivamente. En este caso la tasa binaria será 20/16 = 5/4 mayor.

El bloque debe ser capaz, por tanto, de desligar las tasas binarias a la entrada y salida. Para ello, la solución propuesta en este proyecto consiste en convertir de serie a paralelo la trama, trabajar sobre ella en este formato y posteriormente reinyectar la trama en serie, a la nueva tasa binaria, como muestra la figura 200.

Figura 200. Anexo A. Árbol de Paralelización. Ejemplo. Realización de la operación con conversiones serie y paralelo.

La realización de la operación requiere, por tanto, de un bloque de conversión serie a paralelo y uno paralelo a serie. Entre medias se coloca el bloque que realiza la operación deseada. Este es el mecanismo empleado, por ejemplo, en el bloque de inserción de inserción, como muestra la figura 123.

Los bloques Xilinx® Serie a Paralelo no son capaces de poner en paralelo un número ilimitado de bits. En ocasiones, y en función de la modulación que se quiere emplear, ese límite se sobrepasa. Para solventar ese inconveniente, empleamos una estructura en árbol para poner en paralelo la trama en dos o más niveles (según sea necesario). La trama original se "trocea" y se realiza la operación deseada en bloques que manejar únicamente una parte de la trama.

Para ello, se realiza una primera operación de conversión serie a paralelo, empleando un bloque Xilinx® Serie a Paralelo, destinada a reducir el tamaño de la trama para que sea manejable por los bloques serie paralelo, típicamente entre 4 y 16 bits en paralelo A continuación un bloque Xilinx® Bitbasher despacha cada uno de sus bits por una línea de datos distinta.

A continuación, un segundo bloque Xilinx® Serie a Paralelo, realiza la conversión serie a paralelo de la parte de la trama que le haya sido adjudicada. Este bloque es equivalente al visto para el caso en que la conversión serie a paralelo puede realizarse en un solo paso (figura 199). A continuación se ejecuta la operación deseada y se convierte la trama a serie.

Finalmente, otro bloque Xilinx® Bitbasher concatena los resultados de las distintas líneas (revertiendo el despacho realizado por el Bitbasher anterior), creando una trama en paralelo del mismo número de líneas paralelas empleadas. Finalmente un bloque Xilinx® Paralelo a Serie inyecta la trama en serie al siguiente bloque. El esquema es el que se muestra en la figura 201.

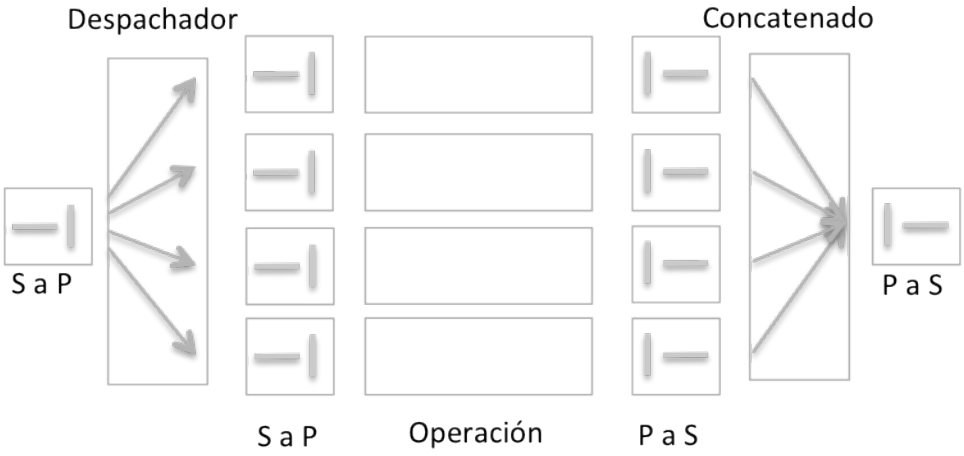

Figura 201. Anexo A. Árbol de Paralelización. Ejemplo. Estructura del árbol.

Para el ejemplo anterior, el proceso sería el que se muestra en la figura 202,en la que se han coloreado ciertos bits para permitir su seguimiento a lo largo de todo el proceso.

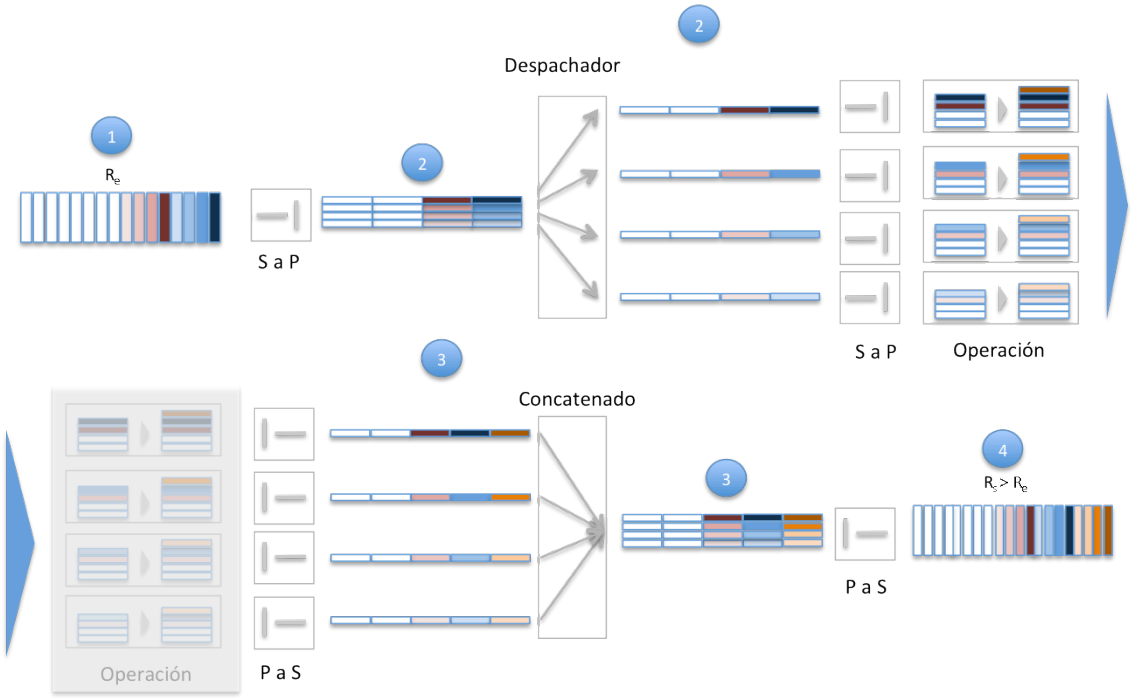

Figura 202. Anexo A. Árbol de Paralelización. Ejemplo. Realización de la operación con conversiones serie y paralelo empleando un árbol .

Se puede apreciar como el primer bloque serie paralelo tiene como objetivo reducir la complejidad de la trama, en este caso en una tasa 1:4. El segundo bloque serie a paralelo tiene como objetivo preparar la parte de la trama que le haya sido asignada para realizar la operación, en este caso con los 4 bits que le han sido asignados.

Cabe destacar que en todos los puntos de control (numerados del 1 al 4) el tiempo de trama es el mismo, modificándose las tasas binarias en cada punto.

Por último, cabe destacar que el proceso de paralelización puede realizarse en cuantas fases sean necesarias hasta conseguir un número de bits que pueda ser manejado por el bloque Xilinx ® Serie a Paralelo de preparación de la trama, que es el bloque que debe manejar un mayor número de bits en paralelo.

10. ANEXO B: FUNCIONES MATLAB® IMPLEMENTADAS

A continuación se adjunta el código en MATLAB® de los bloques Xilinx MCODE empleados en el receptor. Tal y como se explicó (3.5.10 Bloque Filin Modo) anteriormente, estos bloques no entienden el lenguaje MATLAB® completo, sino un subconjunto de dicho lenguaje.

10.1 Randomizer_1trama_v2.m

Este código se emplea en el bloque Aleatorizador.

```
function[data_out]=Randomizer_1trama_v2(data_in , bits_datos_trama)

%Vector de reinicio
%persistent r, r = xl_state([1, 0, 0, 1, 0, 1, 0, 1 ,0 ,0 ,0 ,0 ,0, 0],
{xlUnsigned,1,0});
persistent r,r=xl_state([1, 0, 0, 1, 0, 1, 0, 1 ,0 ,0 ,0 ,0 ,0,
0],{xlUnsigned,1,0}, 15);

%Numero de bits transmitidos en la trama actual y tramas hasta el momento
persistent bits_trx, bits_trx = xl_state(0, {xlUnsigned,12,0});

%Contador
persistent contador, contador = xl_state(0, {xlUnsigned,12,0});

if contador<1055

        if bits_trx<bits_datos_trama
                %Dentro de una trama
                %Se construye la salida
                bit_rand=xfix({xlUnsigned,1,0},xl_xor(r(13),r(14)));
                data_out=xfix({xlUnsigned,1,0},xl_xor(bit_rand,data_in));
                r.push_front_pop_back(bit_rand);
                bits_trx=xfix({xlUnsigned,12,0},bits_trx+1);
                contador=xfix({xlUnsigned,12,0},contador+1);
        else
            %Se acabo la trama unica que tiene en cuenta este codigo
            data_out=0;
            contador=xfix({xlUnsigned,12,0},contador+1);
        end

else %Se reinicia el sistema
    contador=0;
    bits_trx=0;
    data_out=0;
    r(0)=xfix({xlUnsigned,1,0},1);
    r(1)=xfix({xlUnsigned,1,0},0);
    r(2)=xfix({xlUnsigned,1,0},0);
    r(3)=xfix({xlUnsigned,1,0},1);
    r(4)=xfix({xlUnsigned,1,0},0);
    r(5)=xfix({xlUnsigned,1,0},1);
    r(6)=xfix({xlUnsigned,1,0},0);
    r(7)=xfix({xlUnsigned,1,0},1);
    r(8)=xfix({xlUnsigned,1,0},0);
    r(9)=xfix({xlUnsigned,1,0},0);
    r(10)=xfix({xlUnsigned,1,0},0);
    r(11)=xfix({xlUnsigned,1,0},0);
    r(12)=xfix({xlUnsigned,1,0},0);
    r(13)=xfix({xlUnsigned,1,0},0);
    r(14)=xfix({xlUnsigned,1,0},0);
end
```

10.2 Inseción_tailbyte_simple(data_in)

Este código se emplea en el bloque de inserción del byte de cola.

```
function [data_out]= insercion_tailbyte_simple(data_in)

data_out=data_in;
```

10.3 Filtrador_datos.m

Este código se emplea en el bloque .

```
function [data_out]= filtrador_datos(data_in,num_ceros_prev)


%Ojo al numero de ceros previos necesarios segun el tamaÒo de nejec
 persistent nejec, nejec = xl_state(0, {xlUnsigned,12,0});
% persistent cierre, cierre = xl_state(0, {xlUnsigned, 1, 0});


%if cierre==0

if nejec<num_ceros_prev
    data_out=0;
     nejec=nejec+1;
else
    data_out=data_in;
end
```

10.4 Generación_pilotos.m

Este código se emplea en el bloque de inserción de pilotos.

```
function [OutputRE, OutputIM]= pilotos(inputRE, inputIM)



 persistent r, r = xl_state([1,1,1,1,1,1,1,1,1,1,1], {xlSigned,1,0});
 persistent ciclo, ciclo = xl_state(0, {xlSigned,4,0})



 if ciclo > 1

 bit_rand=xl_xor(r(8),r(10));
 PRBS=xl_xor(bit_rand,1);


%  pilotomas_aux=1-2*PRBS;
%  pilotomenos_aux=1-2*xl_not(PRBS);
%  pilotoIM_aux=0;

 if PRBS == 0
 pilotomas=xfix({xlSigned, 2, 0, xlRound, xlWrap}, 1);
 pilotomenos=xfix({xlSigned, 2, 0, xlRound, xlWrap}, -1);
 else
 pilotomas=xfix({xlSigned, 2, 0, xlRound, xlWrap}, -1);
 pilotomenos=xfix({xlSigned, 2, 0, xlRound, xlWrap}, 1);
 end

 pilotoIM=xfix({xlSigned, 2, 0, xlRound, xlWrap}, 0);


 ceropaddingls=xfix({xlSigned, 54, 0, xlRound, xlWrap}, 0);
 ceropaddingMS=xfix({xlSigned, 56, 0, xlRound, xlWrap}, 0);

 zero_DC=xfix({xlSigned, 2, 0, xlRound, xlWrap}, 0);
```

```
portadora10RE=xl_slice(inputRE,383,360 );
portadora9RE=xl_slice(inputRE,359 ,312);
portadora8RE=xl_slice(inputRE,311,264 );
portadora7RE=xl_slice(inputRE,263 ,216 );
portadora6RE=xl_slice(inputRE, 215,192);
portadora5RE=xl_slice(inputRE,191,168);
portadora4RE=xl_slice(inputRE,167,120 );
portadora3RE=xl_slice(inputRE,119,72 );
portadora2RE=xl_slice(inputRE,71, 24);
portadora1RE=xl_slice(inputRE,23,0);

portadora10IM=xl_slice(inputIM,383,360);
portadora9IM=xl_slice(inputIM,359 ,312);
portadora8IM=xl_slice(inputIM,311,264 );
portadora7IM=xl_slice(inputIM,263 ,216 );
portadora6IM=xl_slice(inputIM, 215,192);
portadora5IM=xl_slice(inputIM,191,168);
portadora4IM=xl_slice(inputIM,167,120 );
portadora3IM=xl_slice(inputIM,119,72 );
portadora2IM=xl_slice(inputIM,71 ,24);
portadora1IM=xl_slice(inputIM,23,0);


%Guardas laterales
 OutputRE_aux =
xl_concat(ceropaddingMS,portadora10RE,pilotomas,portadora9RE,pilotomas,portadora8R
E,pilotomenos,portadora7RE,pilotomenos,portadora6RE,zero_DC,portadora5RE,pilotomen
os,portadora4RE,pilotomas,portadora3RE,pilotomenos,portadora2RE,pilotomas,portador
a1RE,ceropaddingls);
 OutputIM_aux =
xl_concat(ceropaddingMS,portadora10IM,pilotoIM,portadora9IM,pilotoIM,portadora8IM,
pilotoIM,portadora7IM,pilotoIM,portadora6IM,zero_DC,portadora5IM,pilotoIM,portador
a4IM,pilotoIM,portadora3IM,pilotoIM,portadora2IM,pilotoIM,portadora1IM,ceropadding
ls);


%Guardas por el centro
%  OutputRE_aux =
xl_concat(portadora10RE,pilotomas,portadora9RE,pilotomas,portadora8RE,pilotomenos,
portadora7RE,pilotomenos,portadora6RE,ceropaddingMS,zero_DC,ceropaddingls,portador
a5RE,pilotomenos,portadora4RE,pilotomas,portadora3RE,pilotomenos,portadora2RE,pilo
tomas,portadora1RE);
%  OutputIM_aux =
xl_concat(portadora10IM,pilotoIM,portadora9IM,pilotoIM,portadora8IM,pilotoIM,porta
dora7IM,pilotoIM,portadora6IM,ceropaddingMS,zero_DC,ceropaddingls,portadora5IM,pil
otoIM,portadora4IM,pilotoIM,portadora3IM,pilotoIM,portadora2IM,pilotoIM,portadora1
IM);
%
OutputRE=xfix({xlSigned, 512, 0, xlRound, xlWrap}, OutputRE_aux);
OutputIM=xfix({xlSigned, 512, 0, xlRound, xlWrap}, OutputIM_aux);



 r.push_front_pop_back(bit_rand);

 else

     ciclo=ciclo+1;

    OutputRE=xfix({xlSigned, 512, 0, xlRound, xlWrap}, 0);
    OutputIM=xfix({xlSigned, 512, 0, xlRound, xlWrap}, 0);
end
```

10.5 Validación_preambulo.m

Este código se emplea en el bloque de inserción de preámbulo.

```
function [OutputRE, OutputIM,flag]= pilotos(inputRE, inputIM)


 persistent ciclo, ciclo = xl_state(0, {xlSigned,16,0})
```

```
if ciclo > (256*22-1)


   OutputRE=inputRE;
   OutputIM=inputIM;
   flag=xfix({xlBoolean}, 1);


else

    ciclo=ciclo+1;
   flag=xfix({xlBoolean}, 0);
   OutputRE=xfix({xlSigned, 2, 0, xlRound, xlWrap}, 0);
   OutputIM=xfix({xlSigned, 2, 0, xlRound, xlWrap}, 0);
end
```

11. ANEXO C: FUNDAMENTOS DE FPGA

Una FGA es un dispositivo semiconductor programable cuyo elemento fundamental es una matriz de Bloques Lógicos Configurables (CLB) conectados a través de interconexiones programables. Esta capacidad de configurar el dispositivo permite su uso para múltiples propósitos, lo que la diferencia de los Circuitos Integrados para Aplicaciones Especificas (ASIC). Estos últimos, como su nombre indica se construyen para realizar una única función específica, para la cual se optimizan.

Las FPGA pueden ser reprogramables, lo que permite reconfigurarlas para distintas funciones. El tipo dominante en la actualidad es el basado en SRAM (Memoria Estática de Acceso Aleatorio), que pueden ser reprogramables. Siguen fabricándose sin embargo FPGA de una única programación, debido a su bajo coste frente a los ASIC.

Los componentes básicos de los que consta una FPGA (se muestran en la figura 203) son:

- CLB: los bloques lógicos configurables son las unidades lógicas básicas de una FPGA. Aunque depende del dispositivo en concreto, un CLB consiste en una matriz conmutable configurable con 4 o 6 entradas, circuitería para realizar selecciones (como multiplexores) y flip-flops (biestables). Esta matriz es muy flexible y se puede configurar para manejar lógica combinatoria, registros de desplazamiento o RAM.

- Interconexiones: las FPGA constan de interconexiones flexibles que unen las señales entre los CLB y hacia y desde los I/O. El encaminamiento de estas conexiones es realizado por el software de diseño empleado.

- IOB: los bancos de entrada/salida se emplean para agrupar la gran cantidad de estándares de entrada/salida que soportan hoy en día las FPGA, los cuales se cuentan por docenas.

- Memoria: la mayoría de las FPGA constan de bloques de memoria RAM, lo cual es una gran ventaja al encontrarse integrada.

- DCM (Gestor del Reloj Digital): bloque empleado para gestionar el reloj digital.

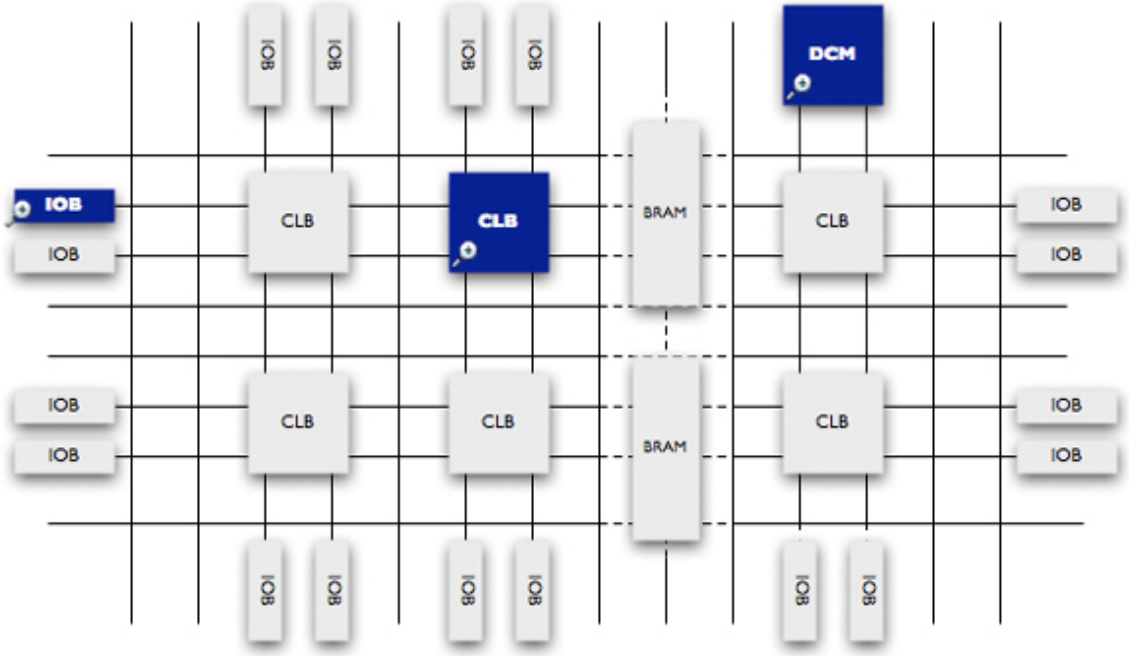

Figura 203. Esquema Básico de bloques de una FPGA. Fuente: Xilinx

La programación de una FPGA se hace mediante un lenguaje de programación específico como HDL. Entre los más conocidos se encuentran: VHDL, Verilog y ABEL. Para reducir el tiempo de desarrollo asociado al empleo de un lenguaje HDL, dada su complejidad, existen opciones de más alto nivel, como el National Instruments LabVIEW que permite una programación gráfica a alto nivel.

Como ya se ha explicado, las FPGA comparten mercado con los ASIC, siendo recomendable el empleo de una u otra tecnología en función de la aplicación. Las mejoras en la velocidad y capacidad de las FPGA, junto a su bajo coste, han provocado que las FPGA se adentren cada vez más en el terreno de las ASIC. Sin embargo, todavía existen campos donde el empleo de ASCI sigue siendo recomendable.

Ventajas de ASCI sobre FPGA:

- Diseño a medida: se adapta totalmente las especificaciones
- Menor coste unitario: para grandes volúmenes
- Menores factores de forma, dado su diseño específico.
- Mayores velocidades de reloj.
- Menor consumo de potencia

Ventajas de FPGA sobre ASCI

- Menor coste unitario, para volúmenes pequeños.
- Posibilidad de reprogramación
- Menor tiempo de puesta en el mercado, al ser menos complejos los procesos de desarrollo.
- Ciclo de diseño más simple, debido al software que gestiona la mayor parte del encaminamiento, emplazamiento y temporización.
- Ciclo de proyecto más predecible, al estar libre de potenciales re-spins y capacidades del sustrato.
- Equipamiento de desarrollo sencillo
- Aumento en la confidencialidad de las placas.
- Posibilidad de reprogramación remota.

Las principales sectores de aplicación de las FPGA son:

- Defensa y sector aeroespacial: procesamiento de imagen, generación de ondas y reconfiguración parcial para SDR (Software Defined Radio).
- Automoción: sistemas de asistencia, confort...
- Retransmisión: video y audio desde el estudio de producción y transmisión hasta el consumidor.
- Consumidor: pantallas digitales de información, redes de trabajo domesticas, auriculares inteligentes...
- Industria, medicina y ciencia: automatización de tareas, control de motores, imágenes médicas de alta calidad.
- Almacenamiento y servidores: soluciones para procesar datos para NAS (Almacenamiento Conectado a Red), SAN (Red de Área de Almacenamiento), servidores, dispositivos de almacenamiento...
- Comunicaciones inalámbricas: RF (Radio Frecuencia), banda base, conectividad, equipamiento inalámbrico, estándares como WCDMA, HSDPA, WiMAX...

- Comunicaciones por cable: soluciones para todos los dispositivos necesarios en el procesamiento de paquetes.

En la actualidad Xilinx (a cuyo co-fundador, Ross Freeman, se le atribuye la invención en 1984 de la FPGA) y Altera constituyen los dos principales fabricantes mundiales de FPGA de propósito general.

12. ANEXO D: FUNDAMENTOS DE MATLAB®

MATLAB®, Matrix Laboratory, es un software matemático orientada a matrices. Consiste en un lenguaje de computación técnico de alto nivel junto a un entorno interactivo

El software es propiedad de The MathWorks, quien comenzó de comercialización y desarrollo en 1984. En la actualidad se encuentra disponible para múltiples plataformas y goza de gran éxito en el campo de ciencia e ingeniería debido a características como:

- Lenguaje de alto nivel para computación técnica
- Entorno de desarrollo para el manejo de código, archivos y datos
- Herramientas interactivas para exploración, diseño y resolución de problemas de forma iterativa
- Funciones matemáticas para algebra líneal, estadística, análisis de Fourier, filtrado, optimización e integración numérica
- Funciones gráficas 2-D y 3-D para la visualización de datos
- Herramientas para diseñar interfaces gráficas a medida
- Funciones para integrar los algoritmos de MATLAB con aplicaciones externas y lenguajes como C, C++, Fortran, Java, COM y Microsoft Excel

MATLAB® se constituye como el núcleo de una familia de herramientas, entre las que destaca Simulink®, tal y como se muestra en la figura __.

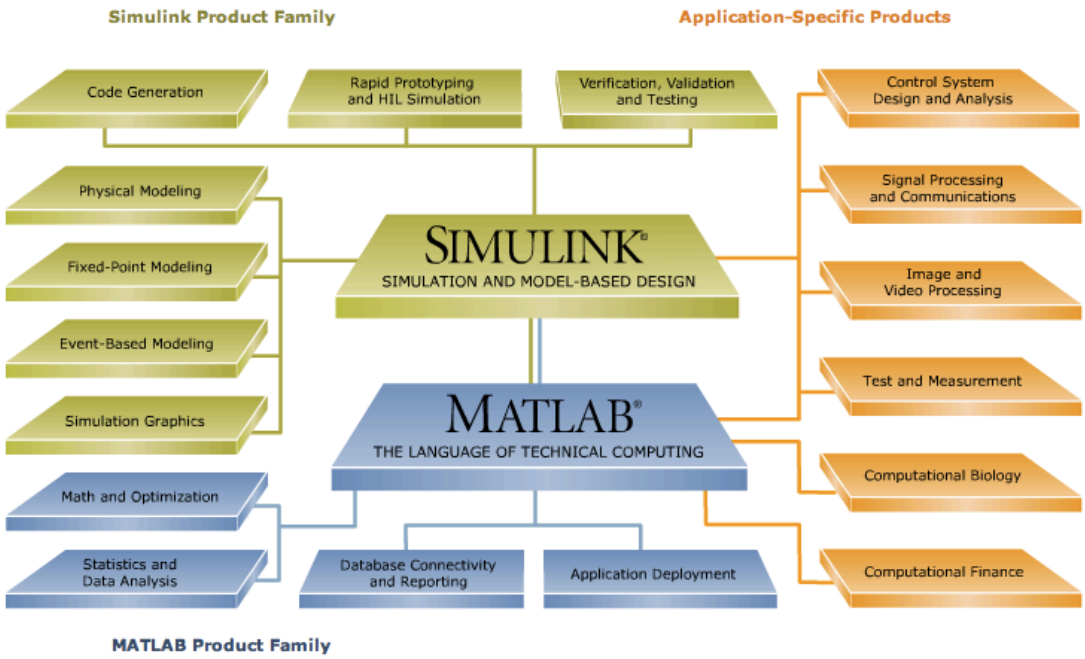

Figura 204. Familia de Productos MATLAB. Fuente: The Mathworks

13. ANEXO E: FUNDAMENTOS DE SIMULINK®

Simulink®, Simulation and Model-Based Design, es una herramienta software consistente en un entorno gráfico interactivo y un conjunto de bibliotecas de bloques personalizables que permiten, mediante un alto nivel de abstracción, diseñar, simular, implementar, evaluar y optimizar sistemas dinámicos (variables en el tiempo)

Tal y como se muestra en la figura 204, del Anexo D, Simulink® se ejecuta sobre MATLAB® y, como este, es desarrollado por The Mathworks y se encuentra disponible para múltiples plataformas.

Entre sus principales características destacan:

- Extensas librerías de bloques predefinidos, con posibilidad de expansión
- Editor gráfico interactivo para ensamblar y gestionar intuitivos diagramas de bloques
- Diseño de sistemas complejos con diferentes niveles de jerarquía mediante el empleo de subsistemas, los cuales aglutinan un conjunto de bloques y señales en un único bloque.
- Exploración de modelos, que permite navegar, crear, configurar y buscar todas las señales, parámetros, propiedades y código generado con el modelo.
- Interfaces de Programación de Aplicaciones, que permiten la conexión con otros programas de simulación e incorporar código escrito a mano,
- Bloques empotrados con Funciones MATLAB® que permiten incorporar algoritmos MATLAB® en Simulink®
- Modos de simulación (Normal, Acelerador, and Acelerador Rápido), para correr simulaciones de forma interpretada o a partir de código compilado C-code. Esta simulación puede hacerse mediante pasos de simulación fijos o variables.
- Debugger gráfico para examinar los resultados de la simulación y diagnosticar el rendimiento y localizar comportamientos no esperados en el diseño.
- Acceso completo a MATLAB® para analizar y visualizar resultados, ajustar el entorno de modelado y definir señales, parámetros y datos de test.

Análisis de modelos y herramientas de diagnóstico para asegurar la consistencia del modelo y localizar errores en el mismo.